MSc Data Mining

Topic 09 : Association Analysis

Part 01 : Association Rule Analysis

Dr Bernard Butler and Dr Kieran Murphy

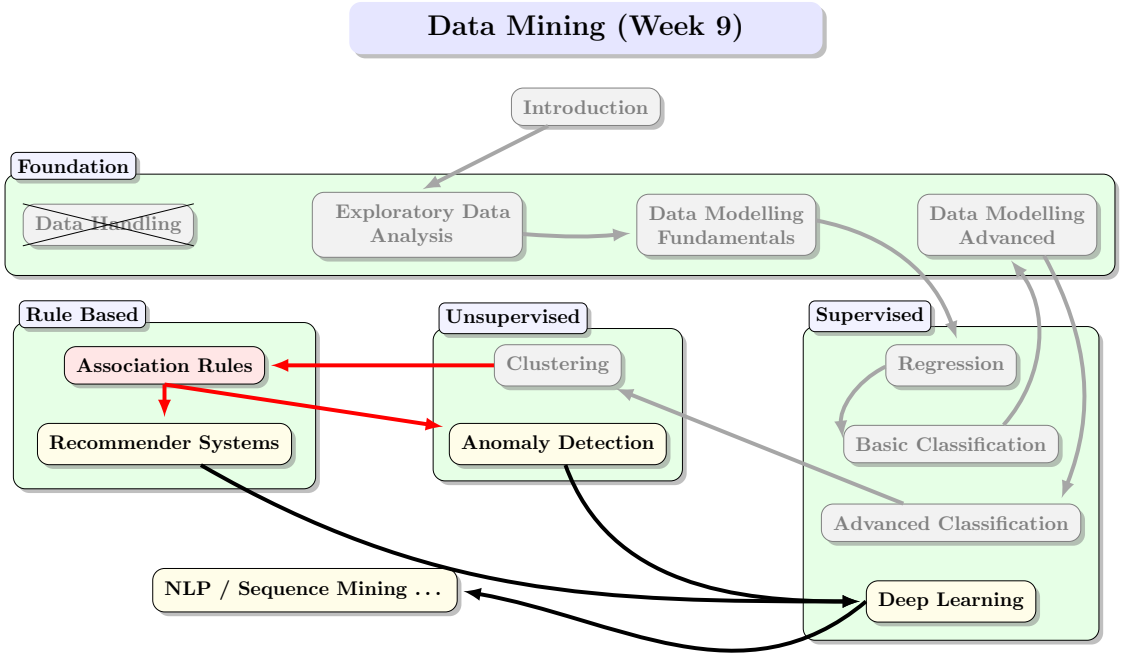
Department of Computing and Mathematics, WIT.

(bernard.butler@wit.ie; kmurphy@wit.ie)

Spring Semester, 2022

Outline

- Fundamental concepts in association rule mining
- Frequent itemset generation
- Rule generation and evaluation
- Chapter 5 of Introduction to Data Mining, by Tan, Steinbach, Karpatne and Kumar


Data Mining (Week 9)
Introduction
Foundation
Data Handling
Exploratory Data Analysis
Data Modelling Fundamentals
Data Modelling Advanced
Rule Based
Association Rules
Recommender Systems
Unsupervised
Clustering
Anomaly Detection
Supervised
Regression
Basic Classification
Advanced Classification
NLP / Sequence Mining ...
Deep Learning

Outline

Association Rule (aka Market Basket) Analysis

The Problem

Given a transaction database:

- A **large** set of items,
  e.g., products sold in a supermarket.
- A **large** set of baskets/transactions,
  each of which is a **small** set of items:
  e.g., the products that a single customer buys on a single shopping trip.

find

- frequent sets of items, called itemsets, that appear together in many transactions.
- "interesting" patterns, associations, correlation or causal structure among sets of items in database.

Interested in connections between items bf not baskets / transactions

Association Rule (aka Market Basket) Analysis

The Problem

Given a transaction database:

- A **large** set of items,
  e.g., products sold in a supermarket.
- A **large** set of baskets/transactions,
  each of which is a **small** set of items:
  e.g., the products that a single customer buys on a single shopping trip.

find

- frequent sets of items, called itemsets, that appear together in many transactions.
- "interesting" patterns, associations, correlation or causal structure among sets of items in database.

Interested in connections between items bf not baskets / transactions

Applications — Market Basket Analysis

Model

item = product

basket = set of products a customer bought in a single shopping trip

Typical Task

Determine products, *A* and *B*, such that knowing a customer bought product *A*, then the probability of them buying product *B* is increased.

Example

An association that has been observed is that people buy beer and diapers* together.

- Is this association interesting?
- How could we utilise this association?
  - Run a sale on diapers and raise price of beer or, run a sale on beer and raise price of diapers.
  - Place beer and diapers near each other, with perhaps something that a customer mightn't buy in between, say salted snacks?

*Beer and Diapers: The Impossible Correlation

Applications — Detecting Plagiarism

Model

item = document containing sentences

basket = sentence

An item (document) is assigned to a basket (sentence) if that document contained that sentence.[†]

Typical Task

Determine items that appear together too often; these are documents that have too many sentences[‡] in common, and could indicate plagiarism.

[†]Notice "in" does not have to match the standard meaning of "in", i.e., items do not have to be "in" baskets. Also, recall, we are interested in connections between items not between baskets.

[‡]or sentence fragments, or spacing/tabbing combinations or 'code clusters' in programs, or ...

Applications — Linking Concepts

Model

item = word

basket = document / web page / tweet / blog

An item (word) is assigned to a basket (document) if that word appears in that document.

Typical Task

Determine items that appear together with high frequency; will identify linked concepts.

Example

Replace ‘document’ by ‘tweet’ and consider Donald Trump’s tweets in run up to 2016 election. Will get different distributions of items depending on whether the tweet was sent from iPhone or Android device.

See Did Trump Tweet It? (didtrumptweetit.com)

Applications — Sentiment Analysis

Model

item = pair of linked pages

basket = web page

An item (pair of linked pages) is assigned to a basket (web page) if that web page is part of that link.

Typical Task

Determine items that appear together with high frequency; pairs of pages with many common links may be about the same concept.

Other Applications

- Services subscribed by the same customer.
- Modules taken by the same student, or interdependent modules in terms of student progression.
- Medications prescribed by a doctor for a patient visit, or underlying conditions.
- Genes that are expressed at the same level.

Scale of Problem

Problem size is measured in terms of two parameters — number of baskets (transactions) and number of items:

Application	**# baskets**	**# items**
Market Basket Analysis		
Our toy dataset	5	6
groceries.csv (R package, arules)	9,835	169
Instacart Dataset (May 2017)	3,421,083	49,688
Amazon (circa 2018)	a very big number	398,040,250
Linking Concepts		
Trump's Tweets	# tweets 100's	# words 1,000's
Identifying webpages with similar concepts	# web pages several billion	# words 100,000,000

Set Theory in 3 Slides — Sets and Elements

Definition 1 (Set)

A set is an unordered collection of distinct well-defined objects (called elements).

- We use braces "$\{$" and "$\}$" to enclose the elements of a set.
- We write $x \in A$ if set A contains element x, and $x \notin A$ otherwise.
  "x is an element of A" "x is not an element of A"
- The empty set, or null set, is denoted by $\{\}$ or $\emptyset$.
- Cardinality/size: number of elements in a set, S is denoted by $|S|$ or $\#S$.

Python supports sets and uses similar notation to mathematics, with one main exception — python represents an empty set using `set()` not `{}`.

Examples

We can define a set by enumerating (listing) its elements:

- Set of decimal digits

$$D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \qquad 1 \in D \qquad 15 \notin D$$

- Set of vowels

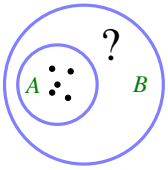
$$V = \{a, e, i, o, u\}$$

Set Theory in 3 Slides — Sets Relations

Equal sets

Two sets, A and B, are equal if they contain the same elements.

Subset and Proper Subset



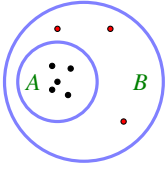
Set A is said to be a subset of B and we write

$$A \subseteq B$$

if and only if every element of A is also an element of B.

If, in addition, B contains **at least one** element not in A we say that A is a proper subset of B, and write

$$A \subset B$$



Subset ($\subseteq$) acts like less than or equals ($\leq$), while proper subset ($\subset$) acts like strictly less than ($<$).
Also have superset ($\supseteq$) and proper superset ($\supset$).

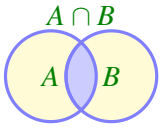
The power set of a set S, denoted by $\mathcal{P}(S)$, is the set of all subsets of S.
A set of size n will have 2^n subsets.

Set Theory in 3 Slides — Sets Operations

Intersection

The intersection of two sets, A and B, denoted by $A \cap B$, is the set that contains all elements that are elements of both A and B. We write

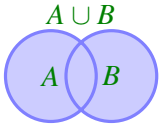
$$A \cap B = \{x \mid (x \in A) \text{ AND } (x \in B)\}$$



Union

The union of two sets, A and B, denoted by $A \cup B$, is the set that contains all elements that are elements of A or B or both. We write

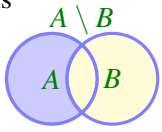
$$A \cup B = \{x \mid (x \in A) \text{ OR } (x \in B)\}$$



Set Difference

The set difference of two sets, A and B, denoted by $A \setminus B$, is the set that contains all elements that are in A but not in B. We write

$$A \setminus B = \{x \mid (x \in A) \text{ AND } (x \notin B)\}$$



Association Rule Mining

Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Transaction Database

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

- Database is a list* of transactions.
- Each transaction is a set†
- Only know if product is in basket — no idea of quantity, selection order.

Sample Association Rules

*A list is an ordered collection (data structure) where duplicates are allowed. (NB: "ordered" $\neq$ "sorted")

†A set is an unordered collection (data structure) where duplicates are not allowed

Association Rule Mining

Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Transaction Database

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

- Database is a list* of transactions.
- Each transaction is a set†
- Only know if product is in basket — no idea of quantity, selection order.

Sample Association Rules

$$\{\text{Diapers}\} \rightarrow \{\text{Beer}\}$$
$$\{\text{Beer}\} \rightarrow \{\text{Diapers}\}$$
$$\{\text{Milk, Bread}\} \rightarrow \{\text{Eggs}\}$$
$$\{\text{Beer}\} \rightarrow \{\text{Bread, Eggs}\}$$

- rules are not symmetric
- obvious $\neq$ interesting

Implication means co-occurrence, not causality!

*A list is an ordered collection (data structure) where duplicates are allowed. (NB: "ordered" $\neq$ "sorted")

†A set is an unordered collection (data structure) where duplicates are not allowed

Definitions — Frequent Itemset

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

Itemset

- A set of one or more items.
- Example: {Milk, Bread, Diapers}
- A k-itemset is an itemset that contains k items.

Support count (σ)

- The number of transactions that contain an itemset.
- Example $\sigma(\{\text{Milk, Bread, Diapers}\}) = 2$ (Only in transactions 4 and 5)

Support (supp)

- Fraction of transactions that contain an itemset.
- Example $\text{supp}(\{\text{Milk, Bread, Diapers}\}) = 2/5 = 0.4$

Frequent Itemset

- An itemset whose support is greater than or equal to a specified minimum support, minsupport, threshold.

Definitions — Association Rule

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

- An association rule is an implication expression of the form

$$A \rightarrow B$$

where

  - A, the antecedent / body / left part
  - B, the consequent / head / right part

are itemsets with no common element (i.e., $A \cap B = \{\}$).

- "if a basket contains all items in set A then it is likely to contain all items in set B"
- Example

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

- We want to find all association rules that are both "important" and "interesting".*

$\Rightarrow$ Need metrics to allow automatic filtering and sorting of rules.

$\Rightarrow$ Need domain specific expertise to evaluate resulting set of rules.

*"Your manuscript is both good and original; but the part that is good is not original, and the part that is original is not good" — Samuel Johnson?

Association Rule Metrics — Support

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

The support of a rule $X \to Y$ is the proportion of transactions that contain all of the items in X and in Y.

So we have

$$\text{supp}(X \to Y) = \frac{\text{Num of transactions with } X \cup Y \text{ as a subset}}{\text{Num of transactions}} = \frac{\sigma(X \cup Y)}{|T|}$$

Examples

- $\text{supp}(\{\text{Bread}\} \to \{\text{Coke}\}) = 1/5 = 0.2 = 20\%$
- $\text{supp}(\{\text{Milk, Diapers}\} \to \{\text{Beer}\}) = 2/5 = 0.4 = 40\%$

Justification

- A rule that has very low support might occur simply by chance.
- A rule with low support is unlikely to be interesting because it might not be profitable to promote items that customers seldom buy together.

Association Rule Metrics — Confidence

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

The confidence of a rule $X \to Y$ is the proportion of transactions that contain Y when we restrict focus to transactions that contain X.
So we have

$$\text{conf}(X \to Y) = \frac{\text{Num of transactions with } X \cup Y \text{ as a subset}}{\text{Num of transactions with } X \text{ as a subset}} = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

Examples

- $\text{conf}(\{\text{Bread}\} \to \{\text{Coke}\}) = 1/4 = 0.25 = 25\%$ $\neq \text{conf}(\{\text{Coke}\} \to \{\text{Bread}\})$
- $\text{conf}(\{\text{Milk, Diapers}\} \to \{\text{Beer}\}) = 2/3 = 0.67 = 67\%$

Justification

- Confidence, measures the reliability of the inference made by a rule.
- The higher the confidence, the more likely it is for Y to be present in transactions that contain X.

Support and Confidence

- Support is symmetrical

$$\text{supp}(X \to Y) = \text{supp}(Y \to X)$$

but, in general, confidence is not symmetrical

$$\text{conf}(X \to Y) \neq \text{conf}(Y \to X)$$

- Usually only interested in association rules with support greater than some minimum threshold, minsupport, and the higher the support the better,
- Usually only interested in association rules with confidence greater than some minimum threshold, minconf, and the higher the confidence the better, However
  - A confidence of 1 might be of no use as it could be constraint of the system, e.g. every mortgage account holder must also have a savings account.
  - A confidence of 0.999 might be very interesting as it could be an indicator of invalid data or fraud.

Transaction Database as a List of Lists/Sets

A transaction database can be represented as list of lists (or sets)

- Easily constructed from a flat file (CSV, one transaction per row)
- Inefficient for filtering.

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

```
1 from pprint import pprint
transactions = [
   line.split(',') for line in open("toy.txt").read().split("\n")
]
pprint(transactions)
```

Searches are linear in number of transactions

... could also use pandas to read CSV and then extract `values` ...

See Practical A where we used this representation (list of sets) exclusively. and Practical B where we started with this representation (list of lists).

```
[['Bread', 'Milk'],
 ['Bread', 'Diapers', 'Beer', 'Eggs'],
 ['Milk', 'Diapers', 'Beer', 'Cola'],
 ['Bread', 'Milk', 'Diapers', 'Beer'],
 ['Bread', 'Milk', 'Diapers', 'Cola']]
```

Transaction Database as Boolean Array (Vertical Format)

A transaction database can be represented as a 2D array of boolean:

TID	Beer	Bread	Cola	Diapers	Eggs	Milk
1	0	1	0	0	0	1
2	1	1	0	1	1	0
3	1	0	1	1	0	1
4	1	1	0	1	0	1
5	0	1	1	1	0	1

- Columns correspond to products.
- Rows correspond to transactions.
- Faster searching, using pandas.
- Array size = number of transactions $\times$ number of products.

$\Rightarrow$ huge array, need to use sparse array data structure, where only store the position of the 1's.

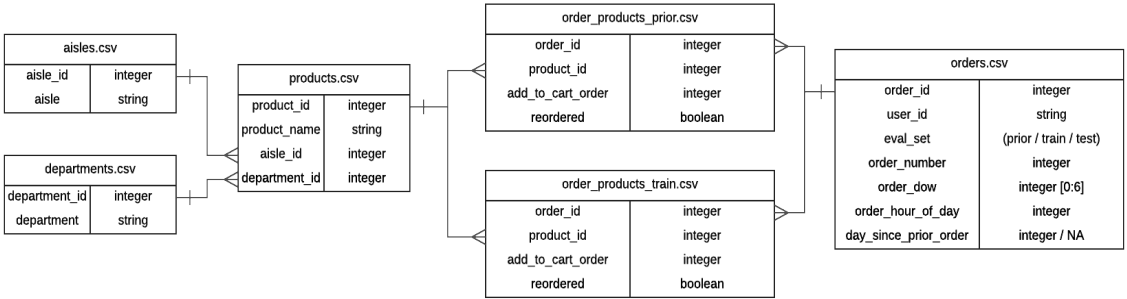
- `mlxtend` library can convert from list of lists representation.

```
2
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_ary = te.fit_transform(transactions,sparse=False)

df = pd.DataFrame(te_ary, columns=te.columns_)
print(df.head())
```

```
    Beer  Bread  Cola  Diapers  Eggs  Milk
0  False   True False    False False  True
1   True   True False     True  True False
2   True  False  True     True False  True
3   True   True False     True False  True
4  False   True  True     True False  True
```

Transaction Database from Other Formats

The Instacart dataset has following ER diagram*



- Structure allows for extra information — sequence products are placed in the basket.
- Need to build list of products in each transaction from `order_products_prior.csv` and `order_products_train.csv`

*https://www.kaggle.com/c/instacart-market-basket-analysis/data

Outline

Formulation of the Association Rule Mining Problem

We want to find all rules with support and confidence above specified thresholds:

Definition 2 (Association Rule Discovery)

Given a set of transactions, T, find all rules $A \rightarrow B$ such that

$$\text{supp}(A \rightarrow B) \geq \text{minsupport} \qquad \text{and} \qquad \text{conf}(A \rightarrow B) \geq \text{minconf}$$

where $A \cup B \subseteq t$ for some $t \in T$.

Brute Force (Enumeration) for mining association rules[†]

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

$\Rightarrow$ Computationally prohibitive expensive because of number of rules.

[†]See Practical A

Formulation of the Association Rule Mining Problem

We want to find all rules with support and confidence above specified thresholds:

Definition 2 (Association Rule Discovery)

Given a set of transactions, T, find all rules $A \rightarrow B$ such that

$$\text{supp}(A \rightarrow B) \geq \text{minsupport} \qquad \text{and} \qquad \text{conf}(A \rightarrow B) \geq \text{minconf}$$

where $A \cup B \subseteq t$ for some $t \in T$.

There exists at least one transaction that contains all the elements that are in set A union set B

Brute Force (Enumeration) for mining association rules[†]

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

$\Rightarrow$ Computationally prohibitive expensive because of number of rules.

[†] See Practical A

Complexity of the Brute Force Method I

"If R is the set of possible associations rules, then $|R|$ is big ..."

Theorem 3

The number of non-trivial association rules, $A \rightarrow B$, that can be constructed from a set of d products is

$$|R| = 3^d - 2^{d+1} + 1$$

Proof

- Each product is either in the antecedent (left / body) of the rule, or the consequent (right / head) of the rule, or not used in the rule. So have

$$|R| = 3 \times 3 \times \cdots \times 3 = 3^d$$

- But this figure incorrectly includes trivial (degenerate) rules where either the antecedent (left) or the consequent (right) is an empty set. We need eliminate these from our tally

$$|R| = 3^d - 2^d - 2^d = 3^d - 2^{d+1}$$

Complexity of the Brute Force Method I

"If R is the set of possible associations rules, then $|R|$ is big ..."

Theorem 3

The number of non-trivial association rules, $A \rightarrow B$, that can be constructed from a set of d products is

$$|R| = 3^d - 2^{d+1} + 1$$

Proof

- Each product is either in the antecedent (left / body) of the rule, or the consequent (right / head) of the rule, or not used in the rule. So have

$$|R| = 3 \times 3 \times \cdots \times 3 = 3^d$$

- But this figure incorrectly includes trivial (degenerate) rules where either the antecedent (left) or the consequent (right) is an empty set. We need eliminate these from our tally

$$|R| = 3^d - 2^d - 2^d = 3^d - 2^{d+1}$$

Complexity of the Brute Force Method I

"If R is the set of possible associations rules, then $|R|$ is big ..."

Theorem 3

The number of non-trivial association rules, $A \rightarrow B$, that can be constructed from a set of d products is

$$|R| = 3^d - 2^{d+1} + 1$$

Proof

- Each product is either in the antecedent (left / body) of the rule, or the consequent (right / head) of the rule, or not used in the rule. So have

$$|R| = 3 \times 3 \times \cdots \times 3 = 3^d$$

- But this figure incorrectly includes trivial (degenerate) rules where either the antecedent (left) or the consequent (right) is an empty set. We need eliminate these from our tally

$$|R| = 3^d - 2^d - 2^d = 3^d - 2^{d+1}$$

Complexity of the Brute Force Method II

Proof (cont)

- But in the previous correction we removed the trivial rule $\{\} \rightarrow \{\}$ twice so we need to increase the count by one to get

$$|R| = 3^d - 2^{d+1} + 1$$

To give this expression some meaning, what does it compute to for our data sets?

Dataset	**Number of products,** d	**Number of rules,** $\lvert R \rvert$
Practical A		
toy.txt	6	602
Practical B		
groceries (arules)	169	4.3×10^{80}
Practical C		
Instacart	49,688	1.58×10^{23708}

Decoupling Frequent Itemset and Rule Generation I

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

Lets look at some of the 602 rules in the toy dataset:

Rule	Support	Confidence
⋮		
{Diapers, Milk} → {Beer}	0.4	0.67
{Beer, Milk} → {Diapers}	0.4	1.00
{Beer, Diapers} → {Milk}	0.4	0.67
{Diapers} → {Beer, Milk}	0.4	0.50
{Milk} → {Beer, Diapers}	0.4	0.50
⋮		

- All the above rules can be obtained by splitting (into antecedent and consequent) the same itemset

  {Beer, Diaper, Milk}

- Rules originating from the same itemset have identical support but can have different confidence.

We can decouple the support and confidence requirements.

Decoupling Frequent Itemset and Rule Generation I

Step 1: Frequent Itemset Generation

- Find all the itemsets that satisfy the minsupport threshold.
- These itemsets are called frequent itemsets.
- Still computationally expensive.
  - Brute force (enumeration) is not as bad[‡], compare $2^d - 1$ vs $3^d - 2^{d+1} + 1$ but still not practical for real datasets
  - Hence methods[§] : Apriori, FP Growth, EClat, ...

Step 2: Rule Generation

- Generate rules from frequent itemsets found in Step 1.
- Extract all the high confidence rules from the frequent itemsets.
- These rules are called strong rules.

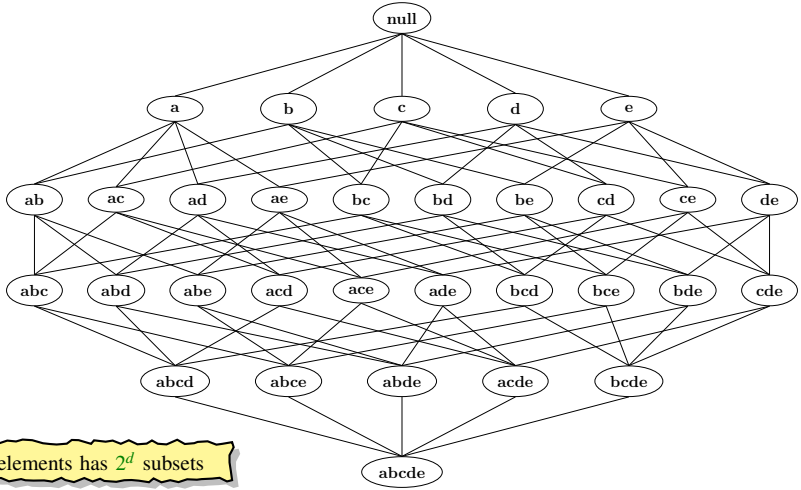
[‡]Compare drowning in water 4 metres deep vs 40 metres.

[§]Comparing Dataset Characteristics that Favor the Apriori, Eclat or FP-Growth Frequent Itemset Mining Algorithms, (arxiv.org/pdf/1701.09042)

Outline

Itemset Lattice Diagram

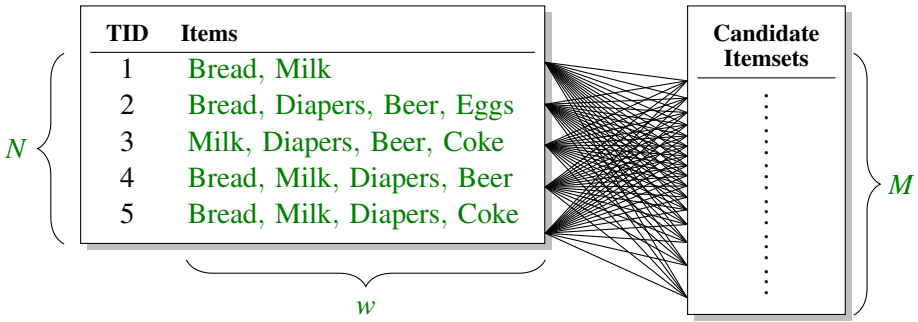
To help in the development of more efficient algorithms we will use the subsets of the set $I = \{a, b, c, d, e\}$, which are shown in the following lattice diagram.



A set of d elements has 2^d subsets

Brute Force

- Each of the $M = 2^d$ itemsets in the lattice is a candidate frequent itemset.
- Count the support of each candidate by scanning the database of N transactions.



- Match each transaction against every candidate

$\Rightarrow$ Complexity is $\mathcal{O}(NMw)$, which is a big deal since $M = 2^d$.

Frequent Itemset Generation Strategies

Reduce the number of candidates (M)

- Complete search = $M = 2^d$
- Use pruning techniques to reduce M.

  We will use the Apriori principle, which relies on the fact that the support never increases when adding an item to an itemset, to eliminate itemsets without having to compute their support.

Reduce the number of transactions (N)

- Reduce size of N as the size of itemset increases
- Used by Direct Hashing and Pruning (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

Reduce the Number of Candidates

Apriori principle

If an itemset is frequent, then all of its subsets must also be frequent

The apriori principle holds due to the anti-monotone property of the support measure:

Definition 4 (Anti-Monotone property)

A metric, f, possess the anti-monotone property if for every itemset, X, that is a proper subset of itemset Y, we have $f(Y) \leq f(X)$, i.e.

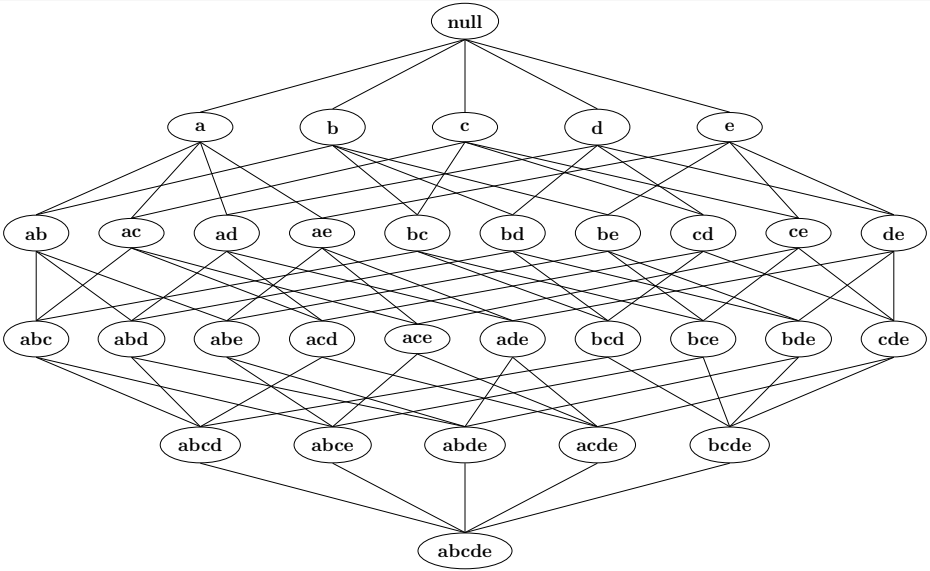
$$(X \subset Y) \implies \text{supp}(X) \geq \text{supp}(Y)$$

for all itemsets X and Y.

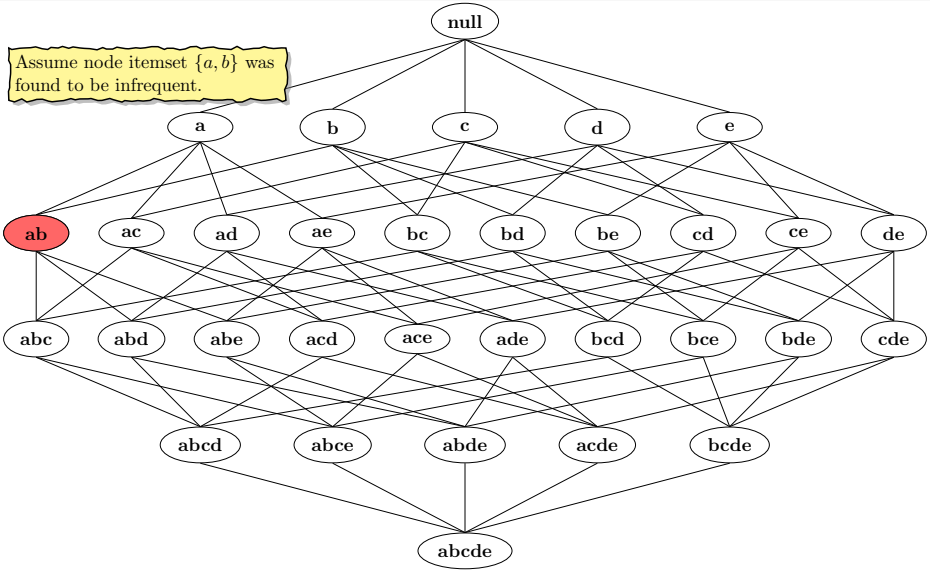
Since support metric has the anti-monotone property this means:

- "*Support of an itemset never exceeds the support of its subsets.*"
- If we find an itemset with support below our threshold, we can ignore it and all its supersets.
- Called support based pruning.

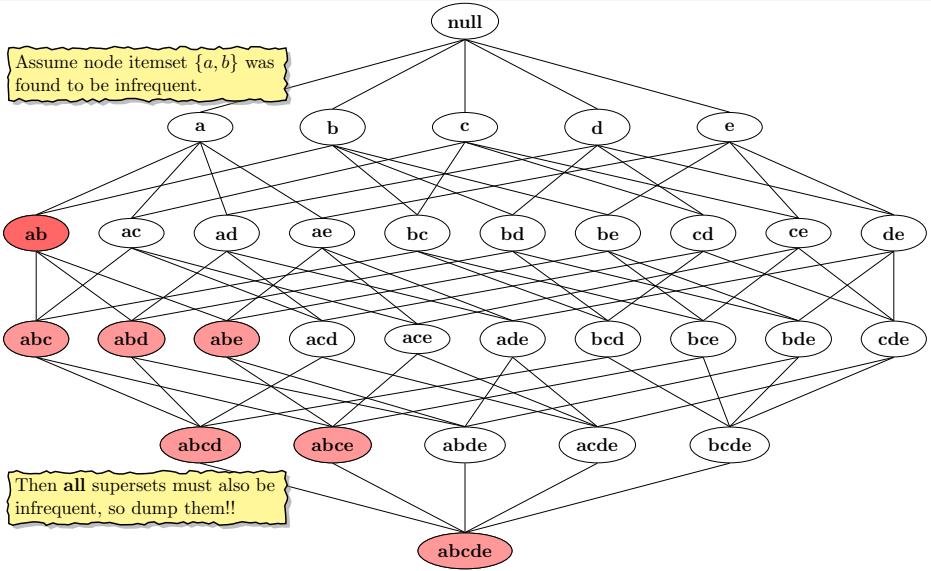
Pruning using the Apriori Principle (Infrequent Itemset)



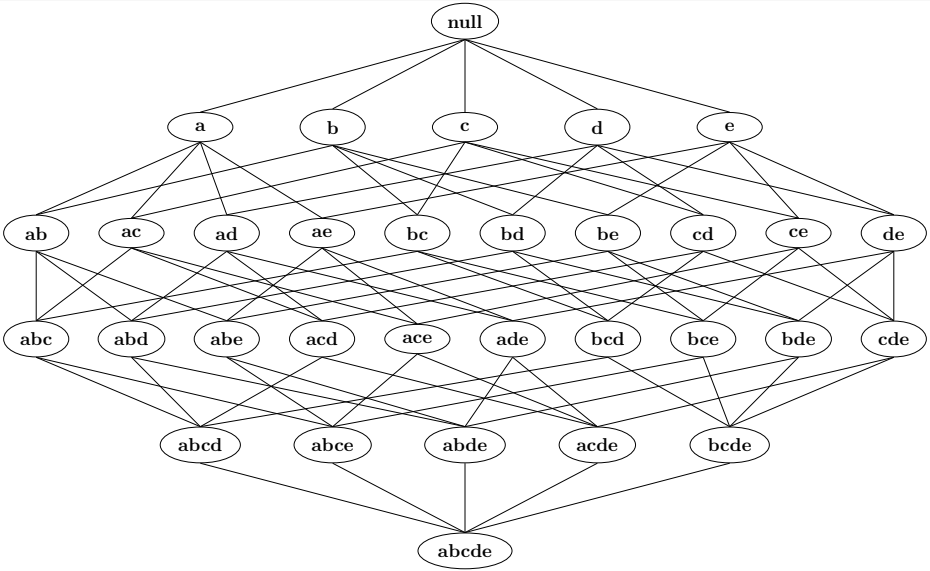
Pruning using the Apriori Principle (Infrequent Itemset)



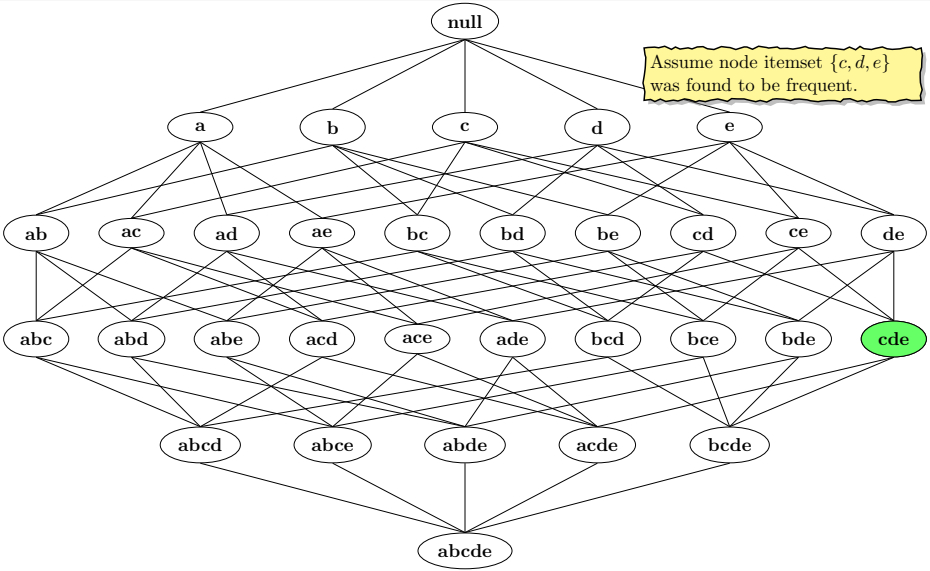
Pruning using the Apriori Principle (Infrequent Itemset)



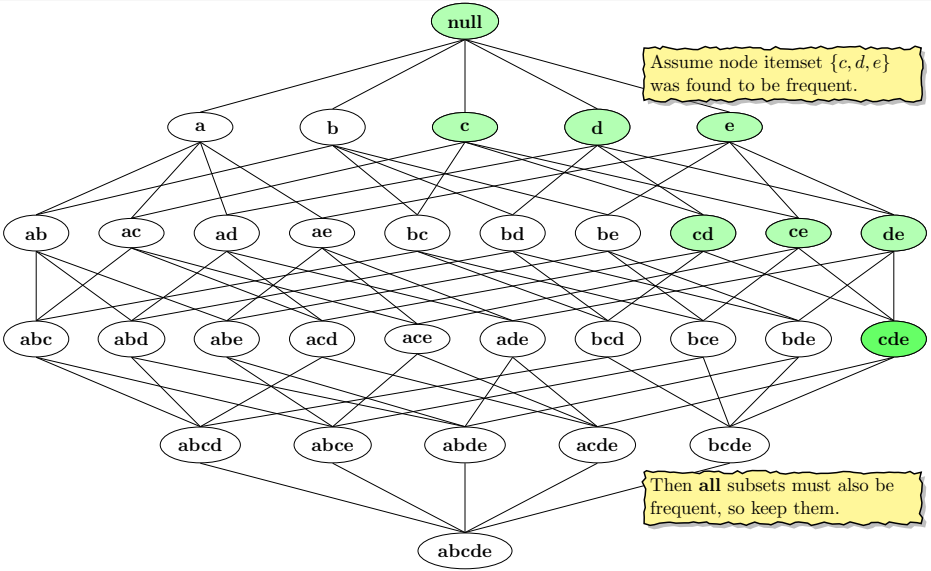
Pruning using the Apriori Principle (Frequent Itemset)



Pruning using the Apriori Principle (Frequent Itemset)



Pruning using the Apriori Principle (Frequent Itemset)



Apriori Algorithm

Initialisation

1 Generate frequent itemsets of length 1

Body

2 For k in $1, 2, 3, 4, 5, \ldots$

3 Generate length $(k+1)$ candidate itemsets from length k frequent itemsets. This is set C_k.

4 Prune candidate itemsets containing subsets of length k that are infrequent.

5 Compute the support of each candidate itemset by scanning the transaction database.

6 Prune candidates that are infrequent, leaving only those that are frequent. This is set F_k.

7 If no new frequent itemsets found, then exit loop.

- Level-wise algorithm — traverses the itemset lattice one level at a time.
- Uses a generate-and-test strategy to determine frequent itemsets.

Apriori Algorithm — Example (using minsupport = 0.6)

Transaction DB

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

Apriori Algorithm — Example (using minsupport = 0.6)

Transaction DB

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

C_1, Candidate 1-Itemsets

Item	Support
Beer	0.6
Bread	0.8
Coke	0.4
Diapers	0.8
Eggs	0.2
Milk	0.8

Apriori Algorithm — Example (using minsupport = 0.6)

Transaction DB

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

F_1, Frequent 1-Itemsets

Item	Support
Beer	0.6
Bread	0.8
Coke	0.4
Diapers	0.8
Eggs	0.2
Milk	0.8

Apriori Algorithm — Example (using minsupport = 0.6)

Transaction DB

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

F_1, Frequent 1-Itemsets

Item	Support
Beer	0.6
Bread	0.8
Coke	0.4
Diapers	0.8
Eggs	0.2
Milk	0.8

C_2, Candidate 2-Itemsets

Item	Support
{Beer, Bread}	0.4
{Beer, Diapers}	0.6
{Beer, Milk}	0.4
{Bread, Diapers}	0.6
{Bread, Milk}	0.6
{Diapers, Milk}	0.6

Apriori Algorithm — Example (using minsupport = 0.6)

Transaction DB

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

F_1, Frequent 1-Itemsets

Item	Support
Beer	0.6
Bread	0.8
Coke	0.4
Diapers	0.8
Eggs	0.2
Milk	0.8

F_2, Frequent 2-Itemsets

Item	Support
{Beer, Bread}	0.4
{Beer, Diapers}	0.6
{Beer, Milk}	0.4
{Bread, Diapers}	0.6
{Bread, Milk}	0.6
{Diapers, Milk}	0.6

Apriori Algorithm — Example (using minsupport = 0.6)

Transaction DB

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

C_3, Candidate 3-Itemsets

Item	Support
{Beer, Bread, Diapers}	
{Beer, Bread, Milk}	
{Beer, Diapers, Milk}	
{Beer, Bread, Milk}	
{Bread, Diapers, Milk}	

F_1, Frequent 1-Itemsets

Item	Support
Beer	0.6
Bread	0.8
Coke	0.4
Diapers	0.8
Eggs	0.2
Milk	0.8

F_2, Frequent 2-Itemsets

Item	Support
{Beer, Bread}	0.4
{Beer, Diapers}	0.6
{Beer, Milk}	0.4
{Bread, Diapers}	0.6
{Bread, Milk}	0.6
{Diapers, Milk}	0.6

Apriori Algorithm — Example (using minsupport = 0.6)

Transaction DB

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

C_3, Candidate 3-Itemsets

Item	Support
{Beer, Bread, Diapers}	
{Beer, Bread, Milk}	
{Beer, Diapers, Milk}	
{Beer, Bread, Milk}	
{Bread, Diapers, Milk}	

F_1, Frequent 1-Itemsets

Item	Support
Beer	0.6
Bread	0.8
Coke	0.4
Diapers	0.8
Eggs	0.2
Milk	0.8

F_2, Frequent 2-Itemsets

Item	Support
{Beer, Bread}	0.4
{Beer, Diapers}	0.6
{Beer, Milk}	0.4
{Bread, Diapers}	0.6
{Bread, Milk}	0.6
{Diapers, Milk}	0.6

Apriori Algorithm — Example (using minsupport = 0.6)

Transaction DB

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

C_3, Candidate 3-Itemsets

Item	Support
{Beer, Bread, Diapers}	
{Beer, Bread, Milk}	
{Beer, Diapers, Milk}	
{Beer, Bread, Milk}	
{Bread, Diapers, Milk}	0.4

F_1, Frequent 1-Itemsets

Item	Support
Beer	0.6
Bread	0.8
Coke	0.4
Diapers	0.8
Eggs	0.2
Milk	0.8

F_2, Frequent 2-Itemsets

Item	Support
{Beer, Bread}	0.4
{Beer, Diapers}	0.6
{Beer, Milk}	0.4
{Bread, Diapers}	0.6
{Bread, Milk}	0.6
{Diapers, Milk}	0.6

Apriori Algorithm — Example (using minsupport = 0.6)

Transaction DB

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

F_3, Frequent 3-Itemsets

Item	Support
{Beer, Bread, Diapers}	
{Beer, Bread, Milk}	
{Beer, Diapers, Milk}	
{Beer, Bread, Milk}	
{Bread, Diapers, Milk}	0.4

F_1, Frequent 1-Itemsets

Item	Support
Beer	0.6
Bread	0.8
Coke	0.4
Diapers	0.8
Eggs	0.2
Milk	0.8

F_2, Frequent 2-Itemsets

Item	Support
{Beer, Bread}	0.4
{Beer, Diapers}	0.6
{Beer, Milk}	0.4
{Bread, Diapers}	0.6
{Bread, Milk}	0.6
{Diapers, Milk}	0.6

Summary

Some issues not covered as beyond scope of this course.

- Efficient techniques to construct the $(k+1)$-itemsets from frequent k-itemsets.
- Techniques to reduce the number of comparisons, by using hashing to group candidate itemsets.
- Alternative algorithms: FP-Growth, ECLAT, etc

See Chapter 5 of *Introduction to Data Mining*, by Tan & Steinbach & Karpatne & Kumar.

Outline

Rule Generation Problem

$B = I \setminus A$

The Problem

Given a frequent itemset I, find all ways to split I into two disjoint, non-empty sets, A and B, i.e.,

$$\underbrace{A \cup B = I}_{I \text{ is split into two}} \qquad \underbrace{A \cap B = \{\}}_{\text{disjoint sets}} \qquad \underbrace{A \neq \{\}, \quad B \neq \{\}}_{\text{both sets are non-empty}}$$

such that the association rule $A \rightarrow B$ has confidence $\geq$ minconf.

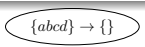
Example

If $I = \{a, b, c, d\}$ is a frequent itemset, the candidate rules are

{abc} → {d}	{abd} → {c}	{acd} → {b}	{bcd} → {a}
{a} → {bcd}	{b} → {acd}	{c} → {abd}	{d} → {acd}
{ab} → {cd}	{ac} → {bd}	{ad} → {bc}	{bc} → {ad}
{bd} → {ac}	{cd} → {ab}		

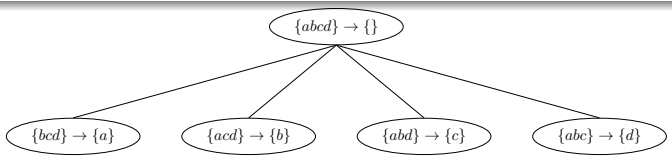
If $I = k$, then there are $2^k - 2$ candidate association rules (ignoring trivial rules $I \rightarrow \{\}$ and $\{\} \rightarrow I$).

Rule Generation with Lattice for Itemset $I = \{a, b, c, d\}$



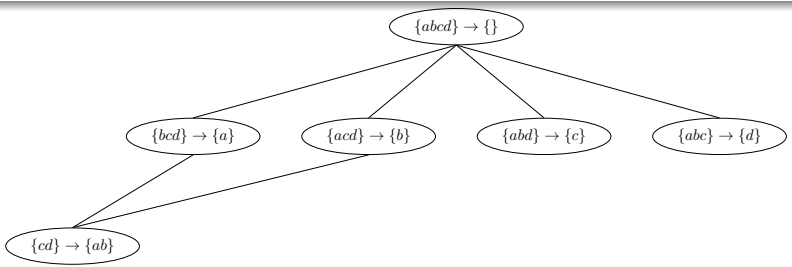
- Start with trivial rule with empty consequent.
- Generate rules by moving one item from antecedent to consequent.
- Generate all subsequent rules by merging two rules:
  - consequent of new rule = union of consequents of original rules.
  - antecedent of new rule = everything not in consequent of new rule.
- Repeat until empty antecedent.

Rule Generation with Lattice for Itemset $I = \{a, b, c, d\}$



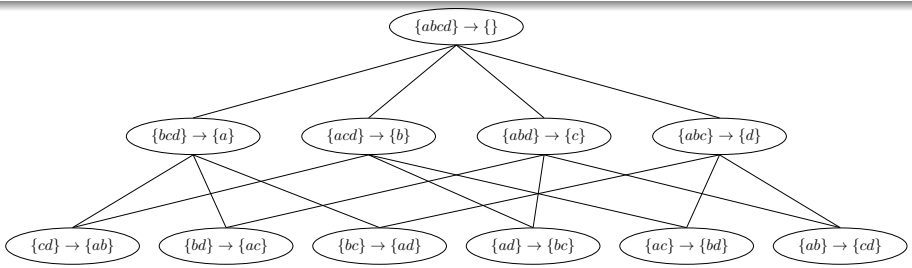
- Start with trivial rule with empty consequent.
- Generate rules by moving one item from antecedent to consequent.
- Generate all subsequent rules by merging two rules:
  - consequent of new rule = union of consequents of original rules.
  - antecedent of new rule = everything not in consequent of new rule.
- Repeat until empty antecedent.

Rule Generation with Lattice for Itemset $I = \{a, b, c, d\}$



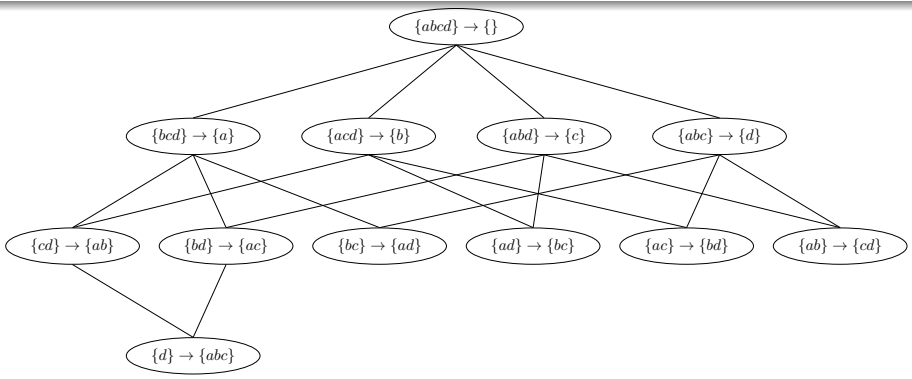
- Start with trivial rule with empty consequent.
- Generate rules by moving one item from antecedent to consequent.
- Generate all subsequent rules by merging two rules:
  - consequent of new rule = union of consequents of original rules.
  - antecedent of new rule = everything not in consequent of new rule.
- Repeat until empty antecedent.

Rule Generation with Lattice for Itemset $I = \{a, b, c, d\}$



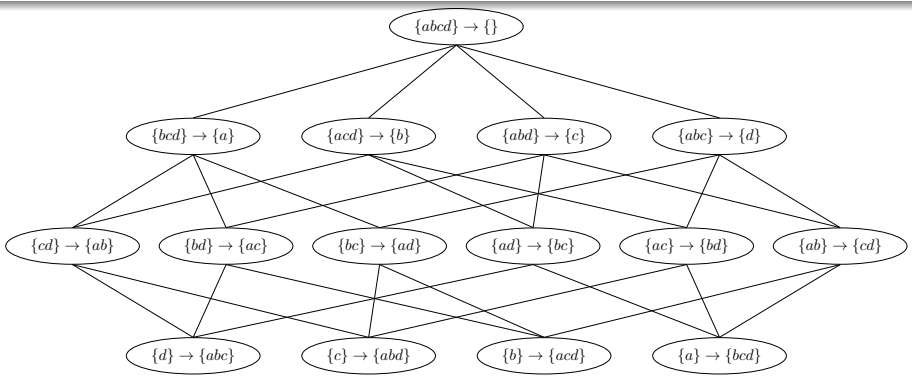
- Start with trivial rule with empty consequent.
- Generate rules by moving one item from antecedent to consequent.
- Generate all subsequent rules by merging two rules:
  - consequent of new rule = union of consequents of original rules.
  - antecedent of new rule = everything not in consequent of new rule.
- Repeat until empty antecedent.

Rule Generation with Lattice for Itemset $I = \{a, b, c, d\}$



- Start with trivial rule with empty consequent.
- Generate rules by moving one item from antecedent to consequent.
- Generate all subsequent rules by merging two rules:
  - consequent of new rule = union of consequents of original rules.
  - antecedent of new rule = everything not in consequent of new rule.
- Repeat until empty antecedent.

Rule Generation with Lattice for Itemset $I = \{a, b, c, d\}$



- Start with trivial rule with empty consequent.
- Generate rules by moving one item from antecedent to consequent.
- Generate all subsequent rules by merging two rules:
  - consequent of new rule = union of consequents of original rules.
  - antecedent of new rule = everything not in consequent of new rule.
- Repeat until empty antecedent.

Pruning in Rule Generation

We would like try to follow a similar approach to that used in frequent itemset selection using support based pruning, but here it is based on confidence rather than support ...

- But, in general, confidence does not have an anti-monotone property:

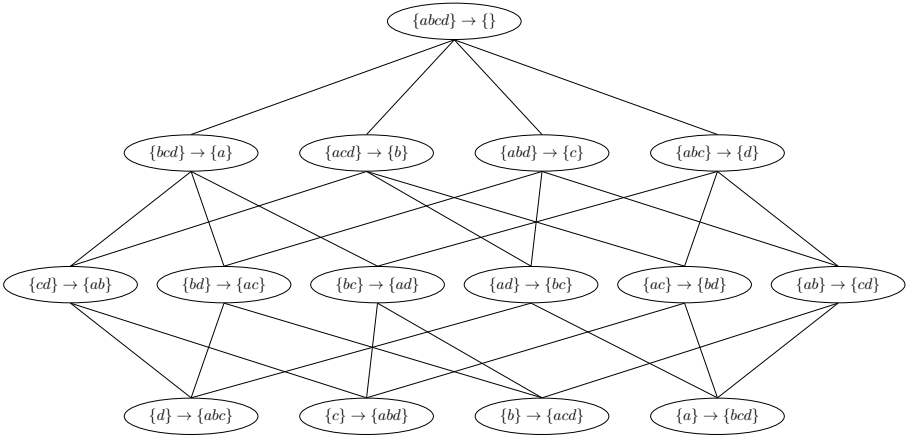
$$\text{conf}(\{\text{abc}\} \rightarrow \{\text{d}\}) \text{ can be larger or smaller than } \text{conf}(\{\text{ab}\} \rightarrow \{\text{d}\})$$

- However confidence of rules generated from **the same itemset** have an anti-monotone property. For example, if $I = \{a, b, c, d\}$ then

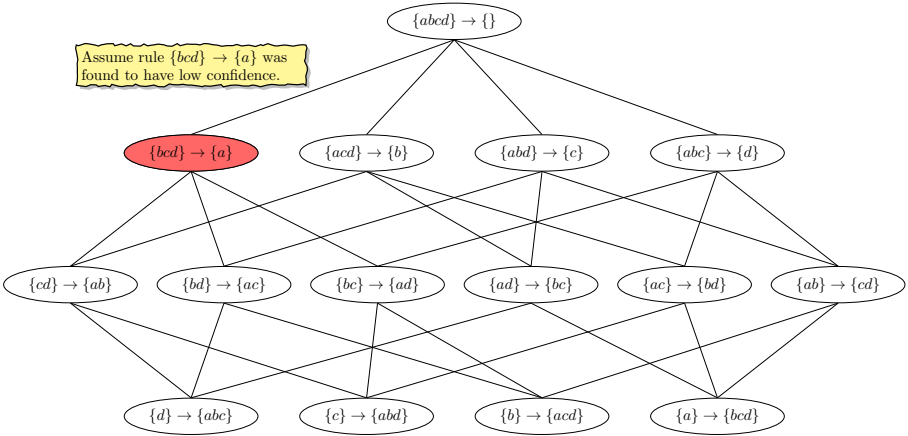
$$\text{conf}(\{\text{abc}\} \rightarrow \{\text{d}\}) \geq \text{conf}(\{\text{ab}\} \rightarrow \{\text{cd}\}) \geq \text{conf}(\{\text{a}\} \rightarrow \{\text{bcd}\})$$

In other words, confidence is non-increasing as items in a rule are moved from the antecedent (left) to the consequent (right).

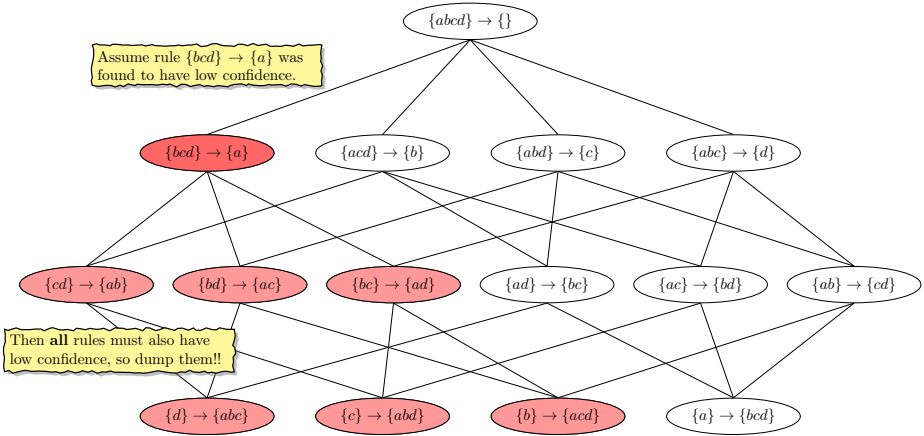
Pruning using the Apriori Principle (Low Confidence)



Pruning using the Apriori Principle (Low Confidence)

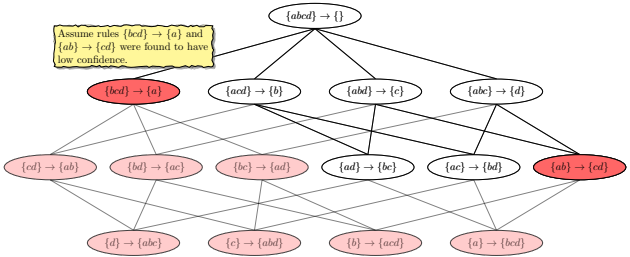


Pruning using the Apriori Principle (Low Confidence)



Candidate Rule Generation Algorithm

- Start with trivial rule with empty consequent.
- Generate rules by moving one item from antecedent to consequent.
- Generate all subsequent rules by merging two rules:
  - consequent of new rule = union of consequents of original rules.
  - antecedent of new rule = everything not in consequent of new rule.
- Prune rule if it or a rule higher up in the lattice has low confidence.
- Repeat until empty antecedent or no new rule with sufficient confidence.



Outline

Rule Evaluation

The Problem

- Association rule algorithms tend to produce too many rules:
  - Many of of the strong rules are uninteresting or redundant.
  - A rule is redundant if $\{abc\} \rightarrow \{d\}$ and $\{ab\} \rightarrow \{d\}$ have the same support and confidence: it does not matter whether $\{c\}$ is included or not.
- **Interestingness** measures can be used to prune/rank the derived patterns

How can we measure the interestingness of a rule?

Subjective measures

A rule (pattern) is interesting if

- it is **unexpected** (surprising to the user); and/or
- **series actionable** (the user can do something with it)

$\Rightarrow$ Need domain expert input.

Objective Measures

Using frequencies compute metrics

- Support
- Confidence or strength
- Lift / Interest or Correlation
- ...

Objective, but no "best" metric

Computing Objective Metrics (using Contingency Table)

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

Given a rule $A \rightarrow B$, the information needed to compute the rule interestingness using objective metrics can be summarised in a contingency table.

A contingency table[¶] stores the count of the transactions which contain, or not, the sets A and/or B.

For example the contingency table for rule $\{\text{Diapers}\} \rightarrow \{\text{Beer}\}$ is

	Diapers	NOT Diapers	sum(row)
Beer	3	0	3
NOT Beer	1	1	2
sum(col)	4	1	5

[¶]Similar to confusion matrix, divide frequencies by overall total to get relative frequencies (probabilities).

Computing Objective Metrics (using Contingency Table)

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

Given a rule $A \rightarrow B$, the information needed to compute the rule interestingness using objective metrics can be summarised in a contingency table.

A contingency table[¶] stores the count of the transactions which contain, or not, the sets A and/or B.

For example the contingency table for rule $\{\text{Diapers}\} \rightarrow \{\text{Beer}\}$ is

	Diapers	NOT Diapers	sum(row)
Beer	3	0	3
NOT Beer	1	1	2
sum(col)	4	1	5

[¶]Similar to confusion matrix, divide frequencies by overall total to get relative frequencies (probabilities).

Computing Objective Metrics (using Contingency Table)

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

Given a rule $A \rightarrow B$, the information needed to compute the rule interestingness using objective metrics can be summarised in a contingency table.

A contingency table[¶] stores the count of the transactions which contain, or not, the sets A and/or B.

For example the contingency table for rule $\{\text{Diapers}\} \rightarrow \{\text{Beer}\}$ is

	Diapers	NOT Diapers	sum(row)
Beer	3	0	3
NOT Beer	1	1	2
sum(col)	4	1	5

$\text{support} = 3/5 = 0.6$
$\text{confidence} = 3/4 = 0.75$

[¶]Similar to confusion matrix, divide frequencies by overall total to get relative frequencies (probabilities).

Contingency Table for Rule $A \rightarrow B$

Rule $A \rightarrow B$

	B	NOT B	
A	f_{TT}	f_{TF}	$f_{T\bullet}$
NOT A	f_{FT}	f_{FF}	$f_{F\bullet}$
	$f_{\bullet T}$	$f_{\bullet F}$	N

Contingency Table for Rule $A \rightarrow B$

Rule $A \rightarrow B$

	B	NOT B	
A	f_{TT}	f_{TF}	$f_{T\bullet}$
NOT A	f_{FT}	f_{FF}	$f_{F\bullet}$
	$f_{\bullet T}$	$f_{\bullet F}$	N

NOT $A = \overline{A}$ etc

divide by number of transactions

Rule $A \rightarrow B$

	B	$\overline{B}$	
A	$\Pr(A \text{ AND } B)$	$\Pr(A \text{ AND } \overline{B})$	$\Pr(A)$
$\overline{A}$	$\Pr(\overline{A} \text{ AND } B)$	$\Pr(\overline{A} \text{ AND } \overline{B})$	$\Pr(\overline{A})$
	$\Pr(B)$	$\Pr(\overline{B})$	1

What is wrong with Confidence?

Consider the rule $\{\text{Tea}\} \rightarrow \{\text{Coffee}\}$ with contingency table

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

Using the formula for confidence we have ...

$$\text{Confidence} = \frac{\text{supp}(\{\text{Tea}\} \cup \{\text{Coffee}\})}{\text{supp}(\{\text{Tea}\})} = \frac{15/100}{20/100} = 0.75$$

Or using contingency table notation ...

$$\text{Confidence} = \frac{f_{TT}}{f_{T\bullet}} = 0.75 = \frac{\Pr(\text{Tea AND Coffee})}{\Pr(\text{Tea})} = \Pr(\text{Coffee}|\text{Tea})$$

Confidence is high (0.75), but probability of a person being a coffee drinker is 0.9. So high confidence rule is misleading here, as tea drinkers are less likely to drink coffee (substitute products, so negative correlation).

In fact $\Pr(\text{Coffee}|\overline{\text{Tea}}) = 0.9375$, i.e., non tea drinkers are more likely to drink coffee.

What is wrong with Confidence?

Consider the rule $\{\text{Tea}\} \rightarrow \{\text{Coffee}\}$ with contingency table

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

Using the formula for confidence we have ...

$$\text{Confidence} = \frac{\text{supp}(\{\text{Tea}\} \cup \{\text{Coffee}\})}{\text{supp}(\{\text{Tea}\})} = \frac{15/100}{20/100} = 0.75$$

Or using contingency table notation ...

$$\text{Confidence} = \frac{f_{TT}}{f_{T\bullet}} = 0.75 = \frac{\Pr(\text{Tea AND Coffee})}{\Pr(\text{Tea})} = \Pr(\text{Coffee}|\text{Tea})$$

Confidence is high (0.75), but probability of a person being a coffee drinker is 0.9. So high confidence rule is misleading here, as tea drinkers are less likely to drink coffee (substitute products, so negative correlation).

In fact $\Pr(\text{Coffee}|\overline{\text{Tea}}) = 0.9375$, i.e., non tea drinkers are more likely to drink coffee.

What is wrong with Confidence?

Consider the rule $\{\text{Tea}\} \rightarrow \{\text{Coffee}\}$ with contingency table

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

Using the formula for confidence we have ...

$$\text{Confidence} = \frac{\text{supp}(\{\text{Tea}\} \cup \{\text{Coffee}\})}{\text{supp}(\{\text{Tea}\})} = \frac{15/100}{20/100} = 0.75$$

Or using contingency table notation ...

$$\text{Confidence} = \frac{f_{11}}{f_{1\bullet}} = 0.75 = \frac{\Pr(\text{Tea AND Coffee})}{\Pr(\text{Tea})} = \Pr(\text{Coffee}|\text{Tea})$$

Confidence is high (0.75), but probability of a person being a coffee drinker is 0.9. So high confidence rule is misleading here, as tea drinkers are less likely to drink coffee (substitute products, so negative correlation).

In fact $\Pr(\text{Coffee}|\overline{\text{Tea}}) = 0.9375$, i.e., non tea drinkers are more likely to drink coffee.

Interest Factor, Lift

Definition

$$\text{lift}(A \to B) = \frac{\text{supp}(A \cup B)}{\text{supp}(A) \times \text{supp}(B)} = \frac{N \times f_{TT}}{f_{T\bullet} \times f_{\bullet T}} = \frac{\Pr(B|A)}{\Pr(B)}$$

Interpretation

$$\text{lift}(A \to B) = \begin{cases} > 1 & \text{if } A \text{ and } B \text{ are positively related} \\ = 1 & \text{if } A \text{ and } B \text{ are independent} \\ < 1 & \text{if } A \text{ and } B \text{ are negatively related} \end{cases}$$

Problems

Rules that hold 100% of the time may not have large lift. For example, consider rule:

$$\{\text{Vietnam veteran}\} \to \{\text{Aged 5 or more}\}$$

If 5% of people are Vietnam veterans and 90% of the people are more than 5 years old, we get a lift of 1/0.9=1.11 which is only slightly above 1 for the rule.

Application

Returning to our rule $\{\text{Tea}\} \rightarrow \{\text{Coffee}\}$ with following contingency table

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

We have

- Confidence $\text{conf}(\{\text{Tea}\} \rightarrow \{\text{Coffee}\}) = 0.75$ ... appears to be a strong rule
- But $\Pr(\text{Coffee}) = 0.9$ ... rule is misleading
- Lift explains why ...

$$\text{lift}(\{\text{Tea}\} \rightarrow \{\text{Coffee}\}) = 0.75/0.9$$

This is < 1, therefore Tea and Coffee are negatively associated.

Conviction of a Rule

Definition

$$\text{conv}(A \rightarrow B) = \frac{\text{supp}(A) \times \text{supp}(\overline{B})}{\text{supp}(A \cup \overline{B})} = \frac{f_{T\bullet} \times f_{\bullet F}}{N \times f_{TF}} = \frac{\Pr(A) \times \Pr(\overline{B})}{\Pr(A \text{ AND } \overline{B})}$$

Interpretation

- Conviction is a measure of the implication and has value 1 if items are unrelated.
- $\text{conv}(A \rightarrow B)$ can be interpreted as the the ratio of the expected frequency that A occurs without B (that is to say, the frequency that the rule makes an incorrect prediction) if A and B were independent, divided by the observed frequency of incorrect predictions
- A conviction value of 1.2 shows that the rule would be incorrect 20% more often (1.2 times as often) if the association between A and B were purely random chance.

Leverage (proposed by Gregory Piatetsky-Shapiro) of a Rule

Definition

$$\begin{aligned}\text{leverage}(A \to B) &= \text{supp}(A \cup B) - \text{supp}(A) \times \text{supp}(B) \\ &= \frac{f_{TT}}{N} - \frac{f_{F\bullet} \times f_{\bullet T}}{N^2} = \Pr(A \text{ AND } B) - \Pr(A) \times \Pr(B)\end{aligned}$$

Interpretation

- The leverage of a rule, $A \to B$, is the **proportion of additional transactions** covered by both the A and B **above the expected** if A and B are independent.
- $\text{leverage}(A \to B) = 0$ if A and B are independent.

Outline

Summary

- We analyse associations between frequent itemsets that occur together in the same basket, not comparing baskets directly
- Typically applied to market basket analysis, but applicable to many domains where co-occurrence/shared features are of interest
- An association rule is an implication from one frequent itemset to another distinct itemset
- Support (symmetrical) and confidence (not symmetrical) are critical measures of "association strength"
- When looking for frequent itemsets, use A Priori principle on *support*: an itemset is frequent if and only if its subsets are too
- For each frequent item set, use a similar A Priori principle on *confidence* in the rule lattice
- Many objective measures of association: support, confidence, lift, conviction, leverage, but can be skewed by highly nonuniform item frequencies.

Relationship with other techniques

technique	consequent is after antecedent	predict consequent for new user
association analysis	no; both in same basket	no; items are aggregated over transactions/users
sequence mining	yes; consequent should be in later basket	no; items are aggregated
recommendation systems	consequent should be in later basket for another user	yes, by definition!

Sequence Mining is not covered in this module. Recommendation Systems will be covered in Week 10.