# ASSOCIATION RULE MINING

# Introduction

- Finding frequent patterns in a database allows to find useful information.
- But it has some limitations→

# Introduction

A **transactional database** ***D***

| Transaction | Items in the transaction |
| --- | --- |
| T1 | {pasta, lemon, bread, orange} |
| T2 | {pasta, lemon} |
| T3 | {pasta, orange, cake} |
| T4 | {pasta, lemon, orange, cake} |

If minsup = 2, then {pasta, cake} is frequent.

**Can we conclude that people who buy pasta will also buy cakes?**

# Association rule

An **association rule** is a rule of the form
$X \rightarrow Y$ where

- $X$ *and* $Y$ are itemsets,
- and $X \cap Y = \emptyset$

**e.g.** {orange, cake} → {pasta}
{lemon, orange} → {pasta}
{pasta} → {bread}
…

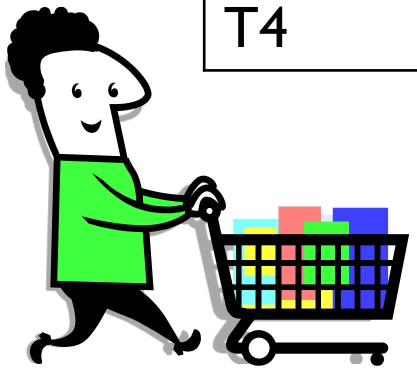
# Support

The **support of a rule** $X \rightarrow Y$ is calculated as

$$sup(X \rightarrow Y) = \frac{sup(X \cup Y)}{|D|}$$

where $|D|$ is the number of transactions.

| Transaction | Items in the transaction |
| --- | --- |
| T1 | {**pasta**, **lemon**, bread, **orange**} |
| T2 | {pasta, lemon} |
| T3 | {pasta, orange, cake} |
| T4 | {**pasta**, **lemon**, **orange**, cake} |



**e.g.** {lemon, orange} ⟶ {pasta}
has a support of 0.5
i.e. two out of four transactions.

# Confidence

The **confidence of a rule** $X \rightarrow Y$ is calculated as

$$conf(X \rightarrow Y) = \frac{sup(X \cup Y)}{sup(X)}.$$

| Transaction | Items in the transaction |
|---|---|
| T1 | {**pasta**, **lemon**, bread, **orange**} |
| T2 | {pasta, lemon} |
| T3 | {pasta, orange, cake} |
| T4 | {**pasta, lemon**, **orange**, cake} |

{lemon, orange} $\longrightarrow$ {pasta} has a confidence of 1.0 (100%)

# Confidence

The **confidence of a rule** $X \rightarrow Y$ is calculated as

$$conf(X \rightarrow Y) = \frac{sup(X \cup Y)}{sup(X)}.$$

| Transaction | Items in the transaction |
|---|---|
| T1 | {**pasta**, **lemon**, bread, orange} |
| T2 | {**pasta**, **lemon**} |
| T3 | {**pasta**, orange, cake} |
| T4 | {**pasta, lemon**, orange, cake} |

{pasta} -> {lemon} has a confidence of 0.75
{lemon} -> {pasta} has a confidence of 1.0

# Association rule mining

**Input**:

- A transaction database (set of transactions)
- A parameter *minsup* $(0 \geq minsup \leq 1)$
- A parameter *minconf* $(0 \geq minconf \leq 1)$

**Output:** each association rule $X \rightarrow Y$ such that:

- $\text{sup}(X \rightarrow Y) \geq minsup$ and
- $\text{conf}(X \rightarrow Y) \geq minconf$

# Example

**minsup = 0.4  minconf = 0.75**

| Transaction | Items in the transaction |
|---|---|
| T1 | {**pasta, lemon**, bread, **orange**} |
| T2 | {pasta, lemon} |
| T3 | {pasta, orange, cake} |
| T4 | {**pasta, lemon**, **orange**, cake} |

- lemon ==> pasta support: 3 confidence: 1
- pasta ==> lemon support: 3 confidence: 0,75
- orange ==> pasta support: 3 confidence: 1
- pasta ==> orange support: 3 confidence: 0,75
- cake ==> pasta support: 2 confidence: 1
- cake ==> orange support: 2 confidence: 1
- lemon orange ==> pasta support: 2 confidence: 1
- orange cake ==> pasta support: 2 confidence: 1
- pasta cake ==> orange support: 2 confidence: 1
- cake ==> pasta orange support: 2 confidence: 1

# Why use the support and confidence?

- With the **support**:
  - find patterns that are less likely to be random.
  - reduce the number of patterns,
  - make the algorithms more efficient.
- The **confidence**:
  - measures the strength of associations
  - obtain an estimation of the conditional probability P( **Y** | X ).
  - **Warning**: a strong association does not mean that there is causality!

# How to find the association rules?

## Naïve approach

1. Create **all** association rules.
2. Calculate their confidence and support by scanning the database.
3. Keep only the valid rules.

This approach is inefficient. For *d* items, there are:

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

possible rules.

For *d* =6, this means 602 rules!
For *d* =100, this means $10^{47}$ rules!
vs itemsets
For *d* =6, this means 63 itemsets

# Observation 1

| Transaction | Items in the transaction |
|---|---|
| T1 | {**pasta, lemon**, bread, **orange**} |
| T2 | {pasta, lemon} |
| T3 | {pasta, orange, cake} |
| T4 | {**pasta, lemon**, **orange**, cake} |

- lemon ==> pasta support: 3 confidence: 1
- pasta ==> lemon support: 3 confidence: 0,75
- orange ==> pasta support: 3 confidence: 1
- pasta ==> orange support: 3 confidence: 0,75

**Observation 1.** All the rules containing the same items can be viewed as having been derived from a same frequent itemset. **e.g.** {pasta, lemon}

# Observation 2

| Transaction | Items in the transaction |
| --- | --- |
| T1 | {**pasta, lemon**, bread, **orange**} |
| T2 | {pasta, lemon} |
| T3 | {pasta, orange, cake} |
| T4 | {**pasta, lemon**, **orange**, cake} |

- lemon ==> pasta support: **3** confidence: **1**
- pasta ==> lemon support: **3** confidence: **0,75**
- orange ==> pasta support: **3** confidence: **1**
- pasta ==> orange support: **3** confidence: **0,75**

**Observation 2.** All the rules containing the same items have the same support, but may not have the same confidence.
**e.g.** {pasta, lemon}

# Observation 3

| Transaction | Items in the transaction |
|---|---|
| T1 | {**pasta**, **lemon**, bread, **orange**} |
| T2 | {pasta, lemon} |
| T3 | {pasta, orange, cake} |
| T4 | {**pasta**, **lemon**, **orange**, cake} |

- lemon ==> pasta support: 3 confidence: 1
- pasta ==> lemon support: 3 confidence: 0,75
- orange ==> pasta support: 3 confidence: 1
- pasta ==> orange support: 3 confidence: 0,75

**Observation 3.** If an itemset is infrequent, all rules derived from that itemset can be ignored. **e.g.** If minsup = 4, rules derived from {pasta, lemon} can be ignored, since its support is 3.

# How to find association rules efficiently?

Aggrawal & Srikant (1993).

**Two steps:**

1. Discover the **frequent itemsets.**
2. Use the frequent itemsets to generate **association rules** having a confidence greater or equal to *minconf*.

Step 1 is the most difficult.

Thus, most studies are on improving the efficiency of Step 1.

# Generating rules

- Each **frequent itemset** X
  of size *k* can produce $2^k$-2 rules.
- A rule can be created by dividing an itemset X in two non empty subsets to obtain a rule $X' \rightarrow Y - X'$.
- Then, the confidence of the rule must be calculated.

# Generating rules

**Example**: using the itemset X={1, 2, 3}, we can generate:

- $\{1,2\} \rightarrow \{3\}$
- $\{1,3\} \rightarrow \{2\}$
- $\{2,3\} \rightarrow \{1\}$
- $\{1\} \rightarrow \{2,3\}$
- $\{2\} \rightarrow \{1,3\}$
- $\{3\} \rightarrow \{1,2\}$

$X \rightarrow Y$ where
$X$ *and* $Y$ are itemsets,
and $X \cap Y = \emptyset$.

# Generating rules

**Example**: using the itemset {a, b, c}, we can generate:

- Generate proper subsets of {a, b, c} :
  - {a}
  - {b}
  - {c}
  - {a, b}
  - {a, c}
  - {b, c}

Use the rule to $X \rightarrow Y$ where
$X$ *and* $Y$ are itemsets,
and $X \cap Y = \emptyset$.
and $X \cup Y =$ {a, b, c}.
to form associations from subsets

- We end up with:
- $\{a\} \rightarrow \{b, c\}$
- $\{b\} \rightarrow \{a, c\}$
- $\{c\} \rightarrow \{a, b\}$
- $\{a, b\} \rightarrow \{c\}$
- $\{a, c\} \rightarrow \{b\}$
- $\{b, c\} \rightarrow \{a\}$

# Generating rules

**Example**: using the itemset {a, b, d, e}: Something like this will be in exams

# Calculating the confidence

**Example**: using the itemset $X=\{1, 2, 3\}$, we can generate:

- $\{1,2\} \rightarrow \{3\}$
- $\{1,3\} \rightarrow \{2\}$
- $\{2,3\} \rightarrow \{1\}$
- $\{1\} \rightarrow \{2,3\}$
- $\{2\} \rightarrow \{1,3\}$
- $\{3\} \rightarrow \{1,2\}$

How can we calculate the confidence of rules derived from X?

- We must know the support of all subsets of **X**.
- We know it already, since if **X** is a frequent itemset, then all its subsets are frequent!

# EVALUATING ASSOCIATIONS

# Evaluating associations

- A large amount of patterns can be discovered
- **How to find the most interesting patterns?**
- Interestingness measures:
  - **objective measures**: statistical reasons for selecting patterns
  - **Subjective measures**: discover surprising or interesting patterns (**e.g.** *diaper* → *beer* is more surprising than *mouse* →*keyboard*).
- It is more difficult to consider subjective measures in the search for patterns.

# Limitations of the support and confidence

If we use a high *minsup* threshold,

- we will find less results,
- it will be faster,
- but we may eliminate some rare patterns that are interesting.

# MINING PATTERNS IN SEQUENCES

# Introduction

- Association rule mining and frequent itemset mining do not consider the time or sequential ordering between events.
- Several techniques to find patterns in one or multiple sequences.

# Sequential pattern mining

## Input:

- A sequence database (a set of sequences)
- A ***minsup*** threshold

## Output:

- All sub-sequences having a support greater or equal to ***minsup***.

**Example**: minsup = 50 %

| ID | sequence |
|---|---|
| 1 | <{a}, {a,b,c} {a, c} {d} {c, f}> |
| 2 | <{a, d}, {c} {b, c} {a, e}> |
| 3 | <{e, f}, {a, b} {d, f} {c}, {b}> |
| 4 | <{e}, {g}, {a, f} {c} {b}, {c}> |

| Pattern | support |
|---|---|
| {a} | 100 % |
| <{a}, {b} > | 100 % |
| <{a, b} > | 50 % |
| … | … |

# Sequential pattern mining

Several algorithms:

- AprioriAll
- GSP (1996)
- PrefixSpan (2001)
- SPADE
- Fast (2011)
- CM-SPAM (2014)

# Sequential rule mining

**Input**:

- A sequence database (a set of sequences)
- A ***minsup*** threshold, a ***minconf*** threshold

**Output**

- Rules of the form X→Y : if the items X appears then items Y will appear after.

| SID | Sequence |
|---|---|
| 1 | $\langle \{a,b\},\{c\},\{f,g\},\{g\},\{e\} \rangle$ |
| 2 | $\langle \{a,d\},\{c\},\{b\},\{a,b,e,f\} \rangle$ |
| 3 | $\langle \{a\},\{b\},\{f,g\},\{e\} \rangle$ |
| 4 | $\langle \{b\},\{f,g\} \rangle$ |

| Pattern | Sup. |
|---|---|
| $\langle \{a\} \rangle$ | 3 |
| $\langle \{a\},\{g\} \rangle$ | 2 |
| $\langle \{a\},\{g\},\{e\} \rangle$ | 2 |
| $\langle \{a\},\{f\} \rangle$ | 3 |
| $\langle \{a\},\{f\},\{e\} \rangle$ | 2 |
| $\langle \{a\},\{c\} \rangle$ | 2 |
| $\langle \{a\},\{c\},\{f\} \rangle$ | 2 |
| $\langle \{a\},\{c\},\{e\} \rangle$ | 2 |
| $\langle \{a\},\{b\} \rangle$ | 2 |
| $\langle \{a\},\{b\},\{f\} \rangle$ | 2 |
| $\langle \{a\},\{b\},\{e\} \rangle$ | 2 |
| $\langle \{a\},\{e\} \rangle$ | 3 |
| $\langle \{a,b\} \rangle$ | 2 |
| $\langle \{b\} \rangle$ | 4 |

| Pattern | Sup. |
|---|---|
| $\langle \{b\},\{g\} \rangle$ | 3 |
| $\langle \{b\},\{g\},\{e\} \rangle$ | 2 |
| $\langle \{b\},\{f\} \rangle$ | 4 |
| $\langle \{b\},\{f,g\} \rangle$ | 2 |
| $\langle \{b\},\{f\},\{e\} \rangle$ | 2 |
| $\langle \{b\},\{e\} \rangle$ | 3 |
| $\langle \{c\} \rangle$ | 2 |
| $\langle \{c\},\{f\} \rangle$ | 2 |
| $\langle \{c\},\{e\} \rangle$ | 2 |
| $\langle \{e\} \rangle$ | 3 |
| $\langle \{f\} \rangle$ | 4 |
| $\langle \{f,g\} \rangle$ | 2 |
| $\langle \{f\},\{e\} \rangle$ | 2 |
| $\langle \{g\} \rangle$ | 3 |
| $\langle \{g\},\{e\} \rangle$ | 2 |

# Periodic pattern mining

- **Periodic Frequent Pattern Mining:** discovering groups of items that appear periodically in a sequence of transactions.
- **Example**:

  {pasta, cookies, orange juice} may be a *frequent periodic pattern* for a particular customer, occurring every week.

# Periodicity of an itemset

## A transaction database

| Transaction | item | |
|---|---|---|
| $T_1$ | {**a**, **c**} | 1 |
| $T_2$ | {e} | 2 |
| $T_3$ | {**a**,b,**c**,d,e} | |
| $T_4$ | {b,c,d,e} | 2 |
| $T_5$ | {**a**,**c**,d} | |
| $T_6$ | {**a**,**c**,e} | 1 |
| $T_7$ | {b,c,e} | 1 |

**Periods of an itemset**:
the number of transactions between each occurrence of the itemset

**Example:**
The periods of the itemset **{a,c}** are: **1,2,2,1,1**

# Periodicity of an itemset

## A transaction database

| Transaction | item |
|---|---|
| $T_1$ | {**a**, **c**} |
| $T_2$ | {e} |
| $T_3$ | {**a**,b,**c**,d,e} |
| $T_4$ | {b,c,d,e} |
| $T_5$ | {**a**,**c**,d} |
| $T_6$ | {**a**,**c**,e} |
| $T_7$ | {b,c,e} |

**Periods of an itemset**:
the number of transactions between each occurrence of the itemset

**Example:**
The periods of the itemset **{a,c}** are: **1,2,2,1,1**

The **maximum periodicity of {a,c}** is **2**

# Definitions of periodic pattern

**Nofong (April 2016):**

**An itemset** X is **periodic** if:

- $(p - p_1) \leq Prd(P^X) - std(P^X)$
- $Prd(X) + std(X) \leq (p + p_1)$

where:

- $p$ is user desired periodicity threshold
- $p_1$ is user desired difference factor
- $Prd(P^X)$ is the mean of $P^X$
- $std(P^X)$ is the standard deviation in $P^X$

Proposed for mining periodic patterns

- **That have similar periodicities**
- **That are positively correlated (not periodic by chance)**

**Fournier-Viger et al (November 2016):**

**An itemset** X is **periodic** if:

- *minAvg ≤ avgper(X) ≤ maxAvg*
- *Minper(X) ≥ minPer*
- *Maxper(X) ≤ maxper*

*where minAvg, maxAvg, minPer, maxPer are parameters set by the user.*

Proposed to give users more flexibility in mining periodic patterns.

# HIGH-UTILITY ITEMSET MINING

# Limitations of frequent itemsets

- **Frequent itemset mining** has **many** applications.
- However, it has important **limitations**
  - many frequent patterns are not interesting,
  - quantities of items in transactions must be 0 or 1
  - all items are considered as equally important (having the same weight)

# High Utility Itemset Mining

- **A generalization of frequent itemset mining:**
  - items can appear more than once in a transaction (**e.g.** a customer may buy 3 bottles of milk)
  - items have a unit profit (**e.g.** a bottle of milk generates $1 of profit)
  - the goal is to find **patterns that generate a high profit**
- **Example**:
  - {peanuts, wine} is a pattern that generates a high profit, although it is rare

# High-utility itemset mining

**Input**

| TID | Transaction |
|---|---|
| $T_1$ | $(a,1),(b,5),(c,1),(d,3),(e,1),(f,5)$ |
| $T_2$ | $(b,4),(c,3),(d,3),(e,1)$ |
| $T_3$ | $(a,1),(c,1),(d,1)$ |
| $T_4$ | $(a,2),(c,6),(e,2),(g,5)$ |
| $T_5$ | $(b,2),(c,2),(e,1),(g,2)$ |

| Item | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ | $g$ |
|---|---|---|---|---|---|---|---|
| Profit | 5 | 2 | 1 | 2 | 3 | 1 | 1 |

***minutil**: a minimum utility threshold set by the user (a positive integer)*

# High-utility itemset mining

## Input

a transaction database

| TID | Transaction |
|---|---|
| $T_1$ | $(a,1),(b,5),(c,1),(d,3),(e,1),(f,5)$ |
| $T_2$ | $(b,4),(c,3),(d,3),(e,1)$ |
| $T_3$ | $(a,1),(c,1),(d,1)$ |
| $T_4$ | $(a,2),(c,6),(e,2),(g,5)$ |
| $T_5$ | $(b,2),(c,2),(e,1),(g,2)$ |

a unit profit table

| Item | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ | $g$ |
|---|---|---|---|---|---|---|---|
| Profit | 5 | 2 | 1 | 2 | 3 | 1 | 1 |

***minutil***: *a minimum utility threshold set by the user* (*a positive integer*)

## Output

All high-utility itemsets (itemsets having a $\text{utility} \geq minutil$)

For example, if *minutil = 33$*, the high-utility itemsets are:

| | |
|---|---|
| {b,d,e} 36$<br>2 transactions | {b,c,d} 34$<br>2 transactions |
| {b,c,d,e} 40$<br>2 transactions | {b,c,e} 37 $<br>3 transactions |

# Utility calculation

a transaction database

| TID | Transaction |
|---|---|
| $T_1$ | $(a,1),(b,5),(c,1),(d,3),(e,1),(f,5)$ |
| $T_2$ | $(b,4),(c,3),(d,3),(e,1)$ |
| $T_3$ | $(a,1),(c,1),(d,1)$ |
| $T_4$ | $(a,2),(c,6),(e,2),(g,5)$ |
| $T_5$ | $(b,2),(c,2),(e,1),(g,2)$ |

a unit profit table

| Item | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ | $g$ |
|---|---|---|---|---|---|---|---|
| Profit | 5 | 2 | 1 | 2 | 3 | 1 | 1 |

The **utility** of the itemset {b,d,e} is calculated as follows:

$$u(\{b,d,e\}) = \underbrace{(5\text{x}2)+(3\text{x}2)+(3\text{x}1)}_{\text{utility in transaction } T_1} + \underbrace{(4\text{x}2)+(2\text{x}3)+(1\text{x}3)}_{\text{utility in transaction } T_2} = \mathbf{36\$}$$

**Challenge**: utility is not anti-monotonic

# How to solve this problem?

- Several algorithms:
  - Two-Phase (PAKDD 2005),
  - IHUP (TKDE, 2010),
  - UP-Growth (KDD 2011),
  - HUI-Miner (CIKM 2012),
  - FHM (ISMIS 2014)
  - EFIM (2015)
  - mHUIMiner (2017)
- Key idea: calculate an upper-bound on the utility of itemsets (e.g. the TWU) that respects the **Apriori property** to be able to prune the search space.

# Conclusion

- We have looked at
  - Association rules
  - Periodic Frequent Patterns
  - Sequential Patterns
  - High Utility Patterns