

Table of contents

1. Introduction

2. Motivation (Text: 6.1)

3. Association Mining: Basic Concepts (Text:6.1)

4. Association Mining: which patterns are interesting? (Text: 6.3)

- 4.1. Lift
- 4.2. Practical Exercise: Association Mining
- 4.3. Chi-square (Text: 6.3.2 and 3.3.2)
- 4.4. Other interestingness measures (Text: 6.3.3)

5. Frequent Itemset Mining (Text 6.1.2)

- 5.1. Apriori algorithm (Text: 6.2.1 and 6.2.3)
- 5.2. Generating Association rules (Text: 6.2.2)
- 5.3. Efficient frequent itemset mining (Text 6.2.6)

6. Advanced Pattern Mining (Text: 6.2.4 and 7)

- 6.1. Practical Exercise: More association mining
- 6.2. Extended data types (Text:7.2.2)
- 6.3. Advanced applications (Text:7.6.2)

7. Practical Exercises

8. Quiz

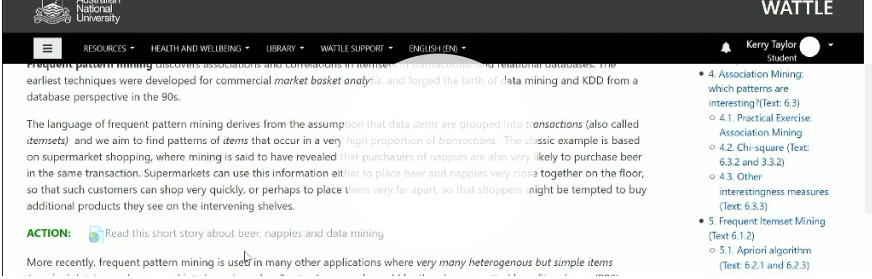
1. Introduction

Most of this material is derived from the text, Han, Kamber and Pei, Chapters 6 and 7, or the corresponding powerpoint slides made available by the publisher. Where a source other than the text or its slides was used for the material, attribution is given. Unless otherwise stated, images are copyright of the publisher, Elsevier.

In this e-book we present the data mining method known as association mining, an early success for the field of data mining. We discuss the types of problems and data for which it is appropriate, the basic methods for evaluation of mining results, and the way the basic algorithm actually works. There are practical exercises using Rattle.

ACTION: If you find it helpful, watch a long lecture on the Association Mining topic. Everything presented in the video is also in the notes.

Association mining lecture 2019



0:00/41:44

CC 1x

讨论 0

扩展

2. Motivation (Text: 6.1)

Frequent pattern mining discovers associations and correlations in itemsets in transactional and relational databases. The earliest techniques were developed for commercial *market basket analysis*, and forged the birth of data mining and KDD from a database perspective in the 90s.

The language of frequent pattern mining derives from the assumption that data *items* are grouped into *transactions* (also called *itemsets*) and we aim to find patterns of *items* that occur in a very high proportion of *transactions*. The classic example is based on supermarket shopping, where mining is said to have revealed that **purchasers** of nappies are also very likely to purchase beer in the same transaction. Supermarkets can use this information either to place beer and nappies very close together on the floor, so that such customers can shop very quickly, or perhaps to place them very far apart, so that shoppers might be tempted to buy additional products they see on the intervening shelves.

ACTION:  [Read this short story about beer, nappies and data mining.](#)

More recently, frequent pattern mining is used in many other applications where *very many heterogenous but simple items (nominal data) may be grouped into large (unordered) sets*. An example could be the pharmaceutical benefits scheme (PBS), where items may be drugs taken under the scheme and each itemset represents the drugs taken by one individual in a fixed period of time. Applications are often but not always commercial, including shopping basket analysis, cross marketing, catalogue design, sale campaign analysis, Web click stream analysis and DNA sequence analysis.

Our work on frequent pattern mining focuses on **association mining** to learn patterns of **association rules**.

3. Association Mining: Basic Concepts (Text:6.1)

Preliminary Definitions

- **item**: an atomic piece of nominal data.
e.g. apple, or orange, or banana.
- **itemset**: A set of one or more items. Let $\mathcal{I} = \{I_1, I_2, \dots, I_m\}$ be an itemset.
e.g. $\{apple, orange, banana\}$
- **k-itemset**: Let $X = \{x_1, \dots, x_k\}$ be an itemset of cardinality k where k is a positive integer. Then X is a k -itemset.
e.g. $\{apple, orange\}$ is a 2-itemset.
- **transaction**: A non-empty itemset, with a unique identifier, in a group called a database. Transaction $T \in D$ where D is a set of **task-relevant transactions** (or a **dataset** or a **database**).
e.g. $D = \{T_1, T_2\}$, transaction $T_1 = \{apple, orange\}$, transaction $T_2 = \{apple, banana, orange\}$
- While each transaction, T has a unique **identifier**, say TID , this is not always explicit in the transaction, but is needed to ensure that multiple transactions containing the same items are distinguished when required.
e.g. $TID(T_1) = 1, TID(T_2) = 2$
- Let A and B be itemsets $A \subseteq \mathcal{I}$ and $B \subseteq \mathcal{I}$
- A transaction T is said to **contain** A if $A \subseteq T$.

Support

The **support count** of itemset A in dataset D is the cardinality of $\{T, \text{ such that } A \subseteq T \text{ and } T \in D\}$. That is, it counts the number of transactions that contain the itemset. Support count is also called *occurrence frequency*, *frequency*, *absolute support* and *count*.

e.g. support count of $\{apple\}$ is 2 in dataset $D = \{\{apple\}, \{apple, orange\}, \{banana, orange\}\}$

The **support** (commonly called **relative support** and sometimes *frequency*) of itemset A in dataset D is the

$$\frac{\text{support count of } A \text{ in } D}{\text{cardinality of } D}.$$

That is, it measures the proportion of transactions that contain it. Let's denote this $support(A)$.

e.g. $support(\{apple\})$ in D above is $2/3$.

An itemset A is **frequent** in dataset D when $support(A) \geq min_sup$ where min_sup is a user-defined, application-specific parameter called the **minimum support threshold**.

Association Rule

An **association rule** is an implication of the form $A \implies B$ where $A \subset \mathcal{I}, B \subset \mathcal{I}, A \neq \emptyset, B \neq \emptyset$ and $A \cap B = \emptyset$.

The rule $A \implies B$ is said to **hold** in the dataset D with **support** s when s is the relative support of $A \cup B$ in D .

The support for an association rule counts the proportion of transactions that contain *every one of both* A 's and B 's items and it does not distinguish between A and B . We can think of support as a measure of significance of the association rule in the dataset -- what proportion of transactions does this rule apply to?

Following a frequentist viewpoint, $s = P(A \cup B)$ i.e. the *probability* that an arbitrary transaction in the dataset contains every item of A and every item of B .

Confidence

The rule $A \implies B$ is said to hold in the dataset D with confidence c when

$$c = \frac{\text{support}(A \cup B)}{\text{support}(A)}.$$

Confidence for an association rule is the support count of $A \cup B$ in D as a proportion of the support count of A in D .

In a frequentist view, this can be interpreted as the conditional probability of B given A i.e. $c = P(B|A)$.

We can think of confidence of the rule as the *truth* of the association rule, that is, what proportion of the transactions that contain A also contain B , or how much does A imply B ?

Strong

An association rule $A \implies B$ is **strong** when $A \cup B$ is **frequent** in D (i.e. satisfies *min_sup*) and the rule also satisfies a minimum confidence parameter, *min_conf*.

That is, $A \implies B$ is strong when $\text{support}(A \cup B) \geq \text{min_sup}$ and also $\frac{\text{support}(A \cup B)}{\text{support}(A)} \geq \text{min_conf}$ where *min_conf* is a user-defined, application-specific parameter called the **minimum confidence threshold**.

ACTION: Do this (paper) exercise:



[Exercise: Basic concepts in frequent pattern mining](#)



[Solution to Exercise: Basic concepts in frequent pattern mining](#)

4. Association Mining: which patterns are interesting? (Text: 6.3)

Strong rules are not necessarily interesting

We need useful, automated measures for assessing pattern *interesting-ness* as there are way too many possible patterns to do it all by human assessment.

Example

Consider the *contingency table*, which compares the frequency of combinations of two items below.

	Drinking	Not Drinking	Sum(row)
Overweight	4000	2000	6000
Not Overweight	3500	500	4000
Sum (column)	7500	2500	10000

Of the 10,000 transactions, 6000 included overweight, 7,500 included drinking, and 4000 included both. The other figures in the table may be derived from those.

Consider $min_sup = 30\%$ and $min_conf = 60\%$ and the *strong* rule $\{overweight\} \implies \{drinking\}$ with *support* 40% and *confidence* 66%

ACTION: Pause and check this rule

The probability of drinking is already 75%, irrespective of overweightness. To suggest that overweightness implies drinking seems misleading at best.

In addition to support and confidence which measure *association*, **lift** is used to evaluate patterns to measure *dependence* amongst itemsets on each side of the rule.

4.1. Lift

In addition to support and confidence which measure *association*, **lift** is used to evaluate patterns to measure *dependence* (also called *correlation*).

Lift

Inspired by probability theory, itemset A is considered independent of itemset B if $P(A \cup B) = P(A) \times P(B)$. Otherwise they are dependent and correlated.

Therefore we define the **lift** of rule $A \implies B$ by $\frac{\text{support}(A \cup B)}{\text{support}(A) \times \text{support}(B)}$.

The numerator is the probability of the customer purchasing both, while the denominator is what the probability of purchasing both *would have been* if the items were independent.

If $\text{lift} = 1$ then A and B are **independent**.

If $\text{lift} > 1$ then they are **positively correlated** as the presence of one suggests the presence of the other. In other words, it suggests that each lifts the likelihood of the other.

If $\text{lift} < 1$ the itemsets are **negatively correlated** and the occurrence of one itemset suggests the more likely absence of the other.

Example

Reconsider the contingency table,

	Drinking	Not Drinking	Sum(row)
Overweight	4000	2000	6000
Not Overweight	3500	500	4000
Sum (column)	7500	2500	10000

What is the lift for $\{\text{overweight}\} \implies \{\text{drinking}\}$ above? $\text{lift} = \frac{4000/10000}{6000/10000 \times 7500/10000} = 16/18 = 0.89$.

Therefore *overweight* and *drinking* are negatively correlated and the rule may be interesting.


ACTION: But wait, lift is defined above by saying A is independent of B when $P(A \cup B) = P(A) \times P(B)$, while probability theory says that if A and B are independent then $P(A \cap B) = P(A) \times P(B)$. Watch the video if you would like this apparent contradiction explained.




[lift and probabilistic independence](#)

4.2. Practical Exercise: Association Mining

ACTION: Get your hands dirty with these exercises!

Answer these questions,  [Association mining Practical: Basics](#) **using the**  [Association Mining Practical: Reference](#) **that explains how to use Rattle to help you.**

Solutions are provided  [Association mining solution: Basics](#) **that could help you if you get stuck. If you use the solutions, make sure you understand them.**

This tutorial video might help you to get started Note that this video covers the subsequent practical exercise in addition to the one right here.

COMP3425/8410 Association Mining



4.3. Chi-square (Text: 6.3.2 and 3.3.2)

For nominal data the correlation between two (and only two) attributes can be measured by the chi-square test. The **chi-square** (χ^2) statistic tests the hypothesis that attributes are independent. For association rules we treat the itemset on each side of the implication arrow as a single binary attribute each, taking on the value of either all the items are present in the transaction, or all the items are absent in the transaction. In this way we consider the correlation amongst the complete itemsets of the left and right sides of a rule.

This chi-square test is also called *Pearson's chi-square test*, not to be confused with [Pearson's r](#) that measures correlation of numerically valued attributes. Both are based on the chi-square distribution.

Chi-square (as normally presented for attribute correlation, from Han et al)

Let A be a nominal that can take on c distinct values a_1, a_2, \dots, a_c and B be a nominal that can take on r distinct values b_1, b_2, \dots, b_r .

Let n be the total number of data tuples (in our case, transactions).

Let (A_i, B_j) represent the case that $A = a_i$ and $B = b_j$

$$\text{Then } \chi^2 = \sum_{i=1}^c \sum_{j=1}^r \frac{(o_{ij} - e_{ij})^2}{e_{ij}},$$

where o_{ij} is the observed frequency (actual count in the data of (A_i, B_j)) and e_{ij} is the expected frequency defined as
$$e_{ij} = \frac{\text{count}(A=a_i) \times \text{count}(B=b_j)}{n}$$

The *degrees of freedom* $(r - 1) \times (c - 1)$ are used to assess the value needed to reject the hypothesis that A and B are independent at some selected significance level. This can be done by a table lookup such as <https://www.itl.nist.gov/div898/handbook/eda/section3/eda3674.htm>

Chi-square interpretation

$\chi^2 = 0$ only if the expected and true number of observations are equal in all cells, that is the attributes are independent.

$p \geq 0.95$ is commonly interpreted as justification for rejecting the null hypothesis that the attributes are independent, that is, for asserting a likely correlation.

In data mining over large datasets, $\chi^2 > 1$ can be used as a rule of thumb to suspect an interesting correlation. But it does not tell you whether that correlation is positive (i.e. they occur together more than expected by chance) or negative (i.e. they occur together less than expected by chance).

Example (original, not from text)

This is most easily expressed in a contingency table. Here we extend the contingency table previously presented for lift [here](#).

The expected value for each value needs to be computed first.

Using the contingency table, this means, e.g.

$$e_{11} = \text{count}(\text{overweight}) \times \text{count}(\text{drinking}) / n = 6000 \times 7500 / 10000 = 4500$$

The contingency table shows expected values indicated in parentheses in each cell.

	Drinking	Not Drinking	Sum(row)
Overweight	4000 (4500)	2000 (1500)	6000
Not Overweight	3500 (3000)	500 (1000)	4000
Sum (column)	7500	2500	10000

Of the 10,000 transactions, 6000 included overweight, 7,500 included drinking, and 4000 included both. The numbers in parenthesis are the expected values.

The chi-square is then

$$\chi^2 = \sum \frac{(\text{observed} - \text{expected})^2}{\text{expected}} = \frac{(4000-4500)^2}{4500} + \frac{(2000-1500)^2}{1500} + \frac{(3500-3000)^2}{3000} + \frac{(500-1000)^2}{1000} = 555.6$$

The degrees of freedom is $(r - 1) \times (c - 1) = (2 - 1) \times (2 - 1) = 1$. Looking up the table given above, the chi-square value needed to reject the hypothesis of independence at the 0.001 significance level is 10.828. Therefore the value of 555.6 suggests the attributes are strongly correlated.

More simply and conveniently, because $\chi^2 > 1$ and the observed value in the top left (4000) is less than the expected value (4500) *overweight* and *drinking* are assumed to be negatively correlated.

Therefore a rule $\{overweight\} \implies \{drinking\}$ may be considered interesting, and this concurs with the assessment using **lift** earlier.

Chi-square (for non-singleton itemsets) (not in text)

To measure the chi-square correlation of itemsets of greater than one item, which is the usual case for association mining, use the 2-D contingency table layout as above, as follows. For itemsets A and B , with n number of transactions, and \bar{A} means the complement of A (ie the itemset containing all the items in all the transactions, except those items in A)

Table 1: Observed contingency table for (A, B)

	B	\bar{B}
A	$n P(A \cap B)$	$n P(A \cap \bar{B})$
\bar{A}	$n P(\bar{A} \cap B)$	$n P(\bar{A} \cap \bar{B})$

4.4. Other interestingness measures (Text: 6.3.3)

Many other measures of interestingness have been proposed in the literature, and can be shown to have intuitively attractive behaviour on at least some itemset patterns.

Here is a definition of many including the support, confidence, lift and chi-square from: Tan, Kumar, Srivastava, *Selecting the right interestingness measure for association patterns*, Proc 8th ACM SIGKDD international conference on Knowledge discovery and data mining, 2002

symbol	measure	range	formula
ϕ	ϕ -coefficient	-1 ... 1	$\frac{P(A,B) - P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
Q	Yule's Q	-1 ... 1	$\frac{P(A,B)P(\bar{A},\bar{B}) - P(A,\bar{B})P(\bar{A},B)}{P(A,B)P(\bar{A},\bar{B}) + P(A,\bar{B})P(\bar{A},B)}$
Y	Yule's Y	-1 ... 1	$\frac{\sqrt{P(A,B)P(\bar{A},\bar{B})} - \sqrt{P(A,\bar{B})P(\bar{A},B)}}{\sqrt{P(A,B)P(\bar{A},\bar{B})} + \sqrt{P(A,\bar{B})P(\bar{A},B)}}$
k	Cohen's	-1 ... 1	$\frac{P(A,B) + P(\bar{A},\bar{B}) - P(A)P(B) - P(\bar{A})P(\bar{B})}{1 - P(A)P(B) - P(\bar{A})P(\bar{B})}$
PS	Piatetsky-Shapiro's	-0.25 ... 0.25	$P(A,B) - P(A)P(B)$
F	Certainty factor	-1 ... 1	$\max\left(\frac{P(B A) - P(B)}{1 - P(B)}, \frac{P(A B) - P(A)}{1 - P(A)}\right)$
AV	added value	-0.5 ... 1	$\max(P(B A) - P(B), P(A B) - P(A))$
K	Klogsen's Q	-0.33 ... 0.38	$\frac{\sqrt{P(A,B)} \max(P(B A) - P(B), P(A B) - P(A))}{\sum_j \max_k P(A_j, B_k) + \sum_k \max_j P(A_j, B_k) - \max_j P(A_j) - \max_k P(B_k)}$
g	Goodman-kruskal's	0 ... 1	$\frac{2 - \max_j P(A_j) - \max_k P(B_k)}{\sum_i \sum_j P(A_i, B_j) \log \frac{P(A_i, B_j)}{P(A_i)P(B_j)}}$
M	Mutual Information	0 ... 1	$\min(-\sum_i P(A_i) \log P(A_i) \log P(A_i), -\sum_i P(B_i) \log P(B_i) \log P(B_i))$
J	J-Measure	0 ... 1	$\max(P(A,B) \log(\frac{P(B A)}{P(B)}) + P(A,\bar{B}) \log(\frac{P(\bar{B} A)}{P(\bar{B})}),$ $P(A,B) \log(\frac{P(A B)}{P(A)}) + P(\bar{A},B) \log(\frac{P(A B)}{P(A)}))$
G	Gini index	0 ... 1	$\max(P(A)[P(B A)^2 + P(\bar{B} A)^2] + P(\bar{A})[P(B \bar{A})^2 + P(\bar{B} \bar{A})^2] - P(B)^2 - P(\bar{B})^2,$ $P(B)[P(A B)^2 + P(\bar{A} B)^2] + P(\bar{B})[P(A \bar{B})^2 + P(\bar{A} \bar{B})^2] - P(A)^2 - P(\bar{A})^2)$
s	support	0 ... 1	$P(A,B)$
c	confidence	0 ... 1	$\max(P(B A), P(A B))$
L	Laplace	0 ... 1	$\max\left(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(B)+2}\right)$
IS	Cosine	0 ... 1	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
γ	coherence(Jaccard)	0 ... 1	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
α	all_confidence	0 ... 1	$\frac{P(A,B)}{\max(P(A), P(B))}$
o	odds ratio	0 ... ∞	$\frac{P(A,B)P(\bar{A},\bar{B})}{P(\bar{A},B)P(A,\bar{B})}$
V	Conviction	0.5 ... ∞	$\max\left(\frac{P(A)P(\bar{B})}{P(A\bar{B})}, \frac{P(B)P(\bar{A})}{P(\bar{B}\bar{A})}\right)$
λ	lift	0 ... ∞	$\frac{P(A,B)}{P(A)P(B)}$
S	Collective strength	0 ... ∞	$\frac{P(A,B) + P(\bar{A}\bar{B})}{P(A)P(B) + P(\bar{A})P(\bar{B})} \times \frac{1 - P(A)P(B) - P(\bar{A})P(\bar{B})}{1 - P(A,B) - P(\bar{A}\bar{B})}$
χ^2	χ^2	0 ... ∞	$\sum_i \frac{(P(A_i) - E_i)^2}{E_i}$

ACTION: Read the paper [Selecting the right objective measure for association analysis](#) from which the following summary of characteristics of interestingness measures is taken.

The property called **null-invariance** is particularly relevant for patterns involving low-frequency itemsets. χ^2 and lift are *not* null-invariant, meaning that their values are significantly raised by all the transactions in which the itemsets do *not* occur. That is they are affected by adding a whole lot of apparently irrelevant transactions to a dataset. On the other hand, as you can see from the table below, confidence is **null-invariant** because its values are affected only by the transactions in which the itemsets *do* occur.

Normally null-invariance is considered a useful property but this may be application-dependent.

Symbol	Measure	Range	P1	P2	P3	O1	O2	O3	O3'	O4
ϕ	ϕ -coefficient	$-1 \dots 0 \dots 1$	Yes	Yes	Yes	Yes	No	Yes	Yes	No
λ	Goodman-Kruskal's	$0 \dots 1$	Yes	No	No	Yes	No	No*	Yes	No
α	odds ratio	$0 \dots 1 \dots \infty$	Yes*	Yes	Yes	Yes	Yes	Yes*	Yes	No
Q	Yule's Q	$-1 \dots 0 \dots 1$	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Y	Yule's Y	$-1 \dots 0 \dots 1$	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
κ	Cohen's	$-1 \dots 0 \dots 1$	Yes	Yes	Yes	Yes	No	No	Yes	No
M	Mutual information	$0 \dots 1$	Yes	Yes	Yes	No**	No	No*	Yes	No
J	J -Measure	$0 \dots 1$	Yes	No	No	No**	No	No	No	No
G	Gini index	$0 \dots 1$	Yes	No	No	No**	No	No*	Yes	No
s	Support	$0 \dots 1$	No	Yes	No	Yes	No	No	No	No
c	Confidence	$0 \dots 1$	No	Yes	No	No**	No	No	No	Yes
L	Laplace	$0 \dots 1$	No	Yes	No	No**	No	No	No	No
V	Conviction	$0.5 \dots 1 \dots \infty$	No	Yes	No	No**	No	No	Yes	No
I	Interest	$0 \dots 1 \dots \infty$	Yes*	Yes	Yes	Yes	No	No	No	No
IS	Cosine	$0 \dots \sqrt{P(A, B)} \dots 1$	No	Yes	Yes	Yes	No	No	No	Yes
PS	Piatetsky-Shapiro's	$-0.25 \dots 0 \dots 0.25$	Yes	Yes	Yes	Yes	No	Yes	Yes	No
F	Certainty factor	$-1 \dots 0 \dots 1$	Yes	Yes	Yes	No**	No	No	Yes	No
AV	Added value	$-0.5 \dots 0 \dots 1$	Yes	Yes	Yes	No**	No	No	No	No
S	Collective strength	$0 \dots 1 \dots \infty$	No	Yes	Yes	Yes	No	Yes*	Yes	No
ζ	Jaccard	$0 \dots 1$	No	Yes	Yes	Yes	No	No	No	Yes
K	Klogen's	$(\frac{2}{\sqrt{3}} - 1)^{1/2} [2 - \sqrt{3} - \frac{1}{\sqrt{3}}] \dots 0 \dots \frac{2}{3\sqrt{3}}$	Yes	Yes	Yes	No**	No	No	No	No

where: P1: $O(\mathbf{M}) = 0$ if $\det(\mathbf{M}) = 0$, i.e., whenever A and B are statistically independent.

P2: $O(\mathbf{M}_2) > O(\mathbf{M}_1)$ if $\mathbf{M}_2 = \mathbf{M}_1 + \begin{bmatrix} k & -k \\ -k & k \end{bmatrix}$.

P3: $O(\mathbf{M}_2) < O(\mathbf{M}_1)$ if $\mathbf{M}_2 = \mathbf{M}_1 + \begin{bmatrix} 0 & k \\ 0 & -k \end{bmatrix}$ or $\mathbf{M}_2 = \mathbf{M}_1 + \begin{bmatrix} 0 & 0 \\ k & -k \end{bmatrix}$.

O1: Property 1: Symmetry under variable permutation.

O2: Property 2: Row and Column scaling invariance.

O3: Property 3: Antisymmetry under row or column permutation.

O3': Property 4: Inversion invariance.

O4: Property 5: Null invariance.

Yes*: Yes if measure is normalized.

No*: Symmetry under row or column permutation.

No**: No unless the measure is symmetrized by taking $\max(M(A, B), M(B, A))$.

ACTION: Pause to check whether support is null-invariant. Can you see why?

5. Frequent Itemset Mining (Text 6.1.2)

Association mining can be viewed as a two-step process

- (1) Find all frequent itemsets--i.e. all the itemsets that occur at least $\text{min_sup} \times \text{cardinality of dataset}$ times in the dataset.
- (2) Use the frequent itemsets to generate strong association rules that satisfy min_conf

The first step is much more computationally demanding than the second, especially when min_sup is low.

5.1. Apriori algorithm (Text: 6.2.1 and 6.2.3)

The *Apriori algorithm* is the archetypical algorithm for finding frequent itemsets.

It implements an iterative level-wise search for frequent itemsets, where the k -itemsets (itemsets of cardinality k) at level k are used to explore the $k+1$ -itemsets at level $k+1$. At each level, the transaction database is scanned to count items in transactions and to collect the items that satisfy minimum support.

Apriori Property

The following observation is used to reduce the search space. This property is very closely related to the *iceberg condition* we saw in data cube materialisation.

All non-empty subsets of a frequent itemset are also frequent

Recall that a frequent itemset is one with at least $\text{least } \text{min_sup}$ support, i.e. one that occurs in at least min_sup transactions. Since each of its subsets occur in all those same transactions (and possibly in additional transactions as well), the support of sub-itemsets cannot be less than min_sup either.

A consequence of this is that **if an itemset is *not* frequent, then any of its supersets cannot be frequent either** (otherwise the apriori property would be violated).

This can be used in a level-wise algorithm like the *apriori algorithm* to reduce the search space, because any $k+1$ -itemset with a non-frequent k -itemset subset (all of which have already been found in the previous iteration) can be ignored.

Apriori algorithm

The basic form of the algorithm is sketched here for reference. You can see the apriori property being used at (5) in procedure `apriori_gen` to prune itemset candidates.

Algorithm: Apriori. Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input:

- D , a database of transactions;
- min_sup , the minimum support count threshold.

Output: L , frequent itemsets in D .

Method:

```
(1)   $L_1 = \text{find\_frequent\_1-itemsets}(D)$ ;  
(2)  for ( $k = 2; L_{k-1} \neq \phi; k++$ ) {  
(3)     $C_k = \text{apriori\_gen}(L_{k-1})$ ;  
(4)    for each transaction  $t \in D$  { // scan  $D$  for counts  
(5)       $C_t = \text{subset}(C_k, t)$ ; // get the subsets of  $t$  that are candidates  
(6)      for each candidate  $c \in C_t$   
(7)         $c.\text{count}++$ ;  
(8)    }  
(9)     $L_k = \{c \in C_k | c.\text{count} \geq min\_sup\}$   
(10) }  
(11) return  $L = \cup_k L_k$ ;  
  
procedure apriori_gen( $L_{k-1}$ :frequent  $(k-1)$ -itemsets)  
(1)  for each itemset  $l_1 \in L_{k-1}$   
(2)    for each itemset  $l_2 \in L_{k-1}$   
(3)      if ( $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2])$   
         $\wedge \dots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ ) then {  
(4)         $c = l_1 \bowtie l_2$ ; // join step: generate candidates  
(5)        if has_infrequent_subset( $c, L_{k-1}$ ) then  
(6)          delete  $c$ ; // prune step: remove unfruitful candidate  
(7)        else add  $c$  to  $C_k$ ;  
(8)      }  
(9)  return  $C_k$ ;  
  
procedure has_infrequent_subset( $c$ : candidate  $k$ -itemset;  
     $L_{k-1}$ : frequent  $(k-1)$ -itemsets); // use prior knowledge  
(1)  for each  $(k-1)$ -subset  $s$  of  $c$   
(2)    if  $s \notin L_{k-1}$  then  
(3)      return TRUE;  
(4)  return FALSE;
```

Many simple further **optimisations** are possible, such as:

- in step (4) of the main procedure above, ignoring transactions from consideration at level k that did not contain any frequent j -itemsets in an earlier level j ($j < k$). This optimisation is another application of the apriori property.
- sampling -- mine a random sample of D instead of the whole dataset

ACTION: Use this example to work through the a-priori algorithm

 [Example: A-Priori algorithm](#)

ACTION: If you need more to understand a-priori, watch this video that works through the same example in detail

 [Explaining the a-priori algorithm](#)

5.2. Generating Association rules (Text: 6.2.2)

Once all frequent itemsets have been found, *strong* association rules are generated as follows.

Note that all frequent itemsets and all their non-empty subsets already satisfy *min_sup* support, so we only need to check *confidence* of the rule, which is $\frac{\text{support}(lhs \cup rhs)}{\text{support}(lhs)}$

For each frequent itemset l ,

 generate all non-empty proper subsets of l

 for each non-empty proper subset s of l

 if $(\text{support_count}(l) / \text{support_count}(s) \geq \text{min_conf})$ then

 output the rule $s \implies (l - s)$

ACTION: Try this (paper) exercise, following the algorithm here, and then check your solution

 [Exercise: Generating association rules from frequent itemsets](#)

 [Solution to Exercise: Generating association rules from frequent itemsets](#)

5.3. Efficient frequent itemset mining (Text 6.2.6)

Recall that if an itemset is frequent, then every subset is also frequent. This enables some significant memory-saving optimisations that can be used in the a-priori and other frequent pattern mining algorithms.

Example:

Consider frequent itemset $A = \{a_1, \dots, a_{100}\}$, so that *min_sup* proportion of transactions contain it. Then all those transactions also contain every non-empty subset of A which are also frequent itemsets. Since there are $2^{100} - 1 \approx 1.27 \times 10^{30}$ non-empty subsets this is a lot.

In data warehouse cuboids, we solved the same problem of combinatorial explosion by looking at closed cells.

Closed

An itemset X is **closed** in dataset D if there is no itemset Y in D such that $X \subset Y$ and Y has the same support count as X in D .

(Note the use of the proper subset relation, not \subseteq here).

Generally, support goes down as an itemset gets bigger, but a closed itemset is as big as it can be without reducing support.

An itemset is a **closed frequent itemset** if it is both closed and frequent.

An itemset X is a **maximal frequent itemset** (or **max-itemset**) in D if X is frequent in D and there is no frequent itemset Y in D such that $X \subset Y$.

A maximal frequent itemset is as big as it can be yet still satisfy the *min_sup* threshold, even though it might have subsets which have higher support but are not maximal-frequent.

The closed frequent itemsets of D , with their support counts, represent all the frequent itemsets of D and their counts without loss of information. The maximal frequent itemsets with their counts, however, do represent all the frequent itemsets, but not their counts, we have only a lower bound on their counts.

Example

Consider D is a dataset of two transactions: $\{\{a_1, \dots, a_{100}\}, \{a_1, \dots, a_{50}\}\}$.

Let *min_sup* be 50%, that is, the minimum support count is 1.

The non-empty frequent itemsets in D are $\{a_1\}, \dots, \{a_{100}\}, \{a_1, a_2\}, \dots, \{a_1, a_{100}\}, \dots, \{a_1, a_2, a_3\}, \dots, \{a_1, \dots, a_{100}\}$, all $2^{100} - 1$ of them.

The closed itemsets of D are $\{a_1, \dots, a_{100}\}$ with support count 1 and $\{a_1, \dots, a_{50}\}$ with support count 2, so they are both frequent.

If we keep track only of all the closed frequent itemsets and their counts, then we can immediately infer all the other frequent itemsets and their counts, by using the maximum count of all the closed frequent itemsets that are its supersets. So $\{a_1\}$ for example, has support count 2. We save a lot of space by not needing to represent all the frequent subsets of the closed frequent itemsets.

The only maximal frequent itemset in D is $\{a_1, \dots, a_{100}\}$ with support count 1.

If we keep track only of all the maximal frequent itemsets and their counts, then we can immediately infer all the other frequent itemsets and a lower bound on their counts, by using the maximum count of all the maximal frequent itemsets each belongs to. So $\{a_1\}$ for example, has support count ≥ 1 . But we save even more space by not needing to represent all the closed frequent subsets of the maximal frequent itemsets.

ACTION: Consider the following worked example showing how this works



[closed and maximal - explained](#)



[Closed and maximal itemsets](#)

6. Advanced Pattern Mining (Text: 6.2.4 and 7)

Algorithms other than a-priori have been developed to reduce database scans or reduce main memory usage and these may scale better over large datasets.

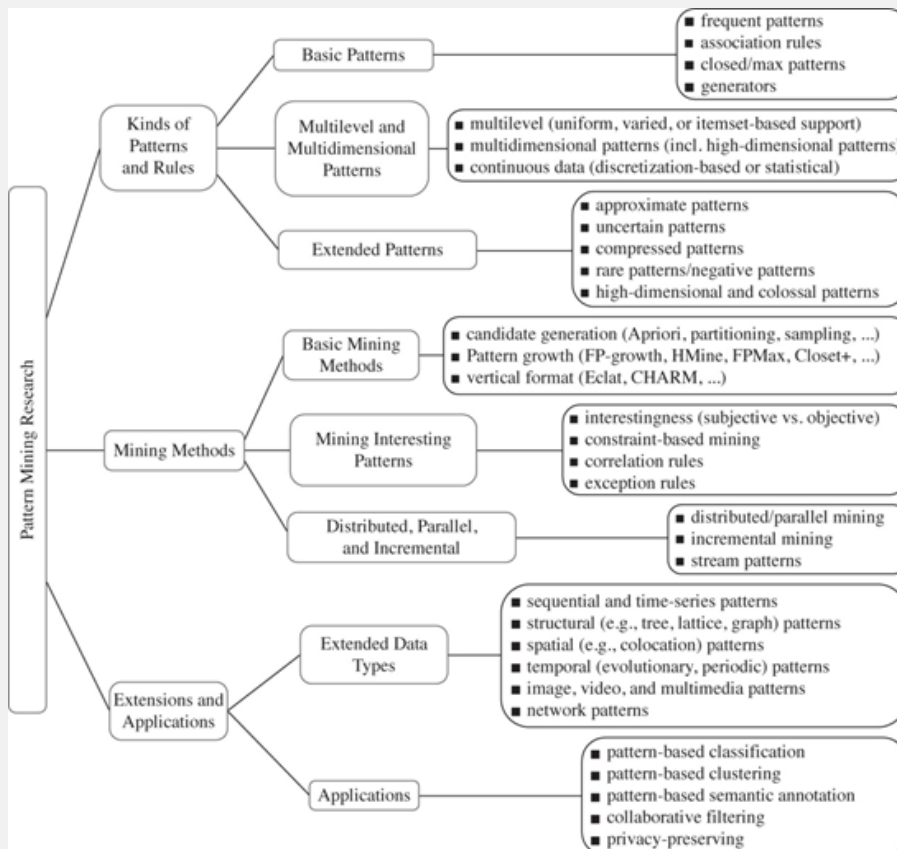
Some of these are:

- FP-growth, frequent pattern growth
- Eclat, equivalence class transformation
- Mining closed and max-patterns

Some algorithms rely on representing transactions in a *vertical* form as <item, set of TIDs> instead of the *horizontal* form <TID, set of items> that we saw previously. However, many of these alternative approaches use the apriori property as does the apriori algorithm.

Advanced Pattern Mining

So far, we have spoken of very simple pattern mining where all the data in a transaction corresponds to multiple values drawn from a single nominal domain. Many extensions have been developed that apply more structure over the input data handled and the rules produced. The following diagram summarises these extensions, including extended sophistication of mining algorithm methods.



6.1. Practical Exercise: More association mining

ACTION: Try your hand at these slightly more extended exercises for association mining with Rattle.

Answer these questions, [!\[\]\(2e897e890e69d81eae4503a8342c36b0_img.jpg\) Association mining Practical: Extended](#) using the [!\[\]\(ce4e2504c7100a62a9a9496b2e01b6e4_img.jpg\) Association Mining Practical: Reference](#) that explains how to use Rattle to help you.

Sample answers are given [!\[\]\(bd1a142de767a21e5362c595f844a4ff_img.jpg\) Association mining Solution: Extended](#) but you really need to be able to do this on your own.

This tutorial video might help to get you started. Note that this video covers the previous practical exercise in addition to the one here.

COMP3425/8410 Association Mining



6.2. Extended data types (Text:7.2.2)

Until now, we have assumed that all *items* are pretty much the same, i.e. they can all be seen as alternative values of a single nominal variable (say, *item*) without any further structure. We can view the problem as single-dimensional analysis over the dimension *item*, although multiple *items* occur in a transaction. This assumption seriously limits the scope of application of association mining.

Multi-dimensional patterns over nominal data

[Earlier](#), we mixed (controllable) lifestyle attributes with disease attributes for a person as indistinguishable items for a **single-dimensional rule**. Such rules are also called **Boolean association rules** since the binary presence or absence of an item is represented. Some advanced pattern mining methods rely on explicitly distinguishing the dimensions for **multidimensional association rules**, but the apriori algorithm does not do this.

A simple solution to distinguish the attributes for single-dimensional algorithms like a-priori is to transform the values of nominal attributes to explicit attribute-value pairs and so to transfer all distinct nominal domains to the one nominal item domain as required by a-priori.

Example

For the domains *Disease* = (*diabetes, cerebrovascular, liver*), and *Lifestyle* = (*drinking, overweight, smoking, exercise*) a transaction such as {*diabetes, overweight, exercise*} would be transformed to the transaction {*Disease=diabetes, Lifestyle=overweight, Lifestyle=exercise*} and these can then be treated homogeneously as if from a single domain.

This transformation need not influence the data mining algorithm, that is, a single-dimensional algorithm like a-priori can be used. It could, however, affect the measures for interestingness of rules and how you would like results to be presented.

Ordinal data

Typically, frequent itemset mining does not take account of any meaningful order amongst the values for categorical data. Therefore ordinal data can be treated as nominal data (as above), or alternatively, where the number of alternative values is very large, as continuous data (as below).

Quantitative (continuous) data

A **quantitative association rule** is one where, based on items that take on **quantitative values** drawn from a numeric domain, rules include attribute values partitioned into discrete intervals, such as age *18-25*. There are two broad approaches that can be used: *static*, or the more sophisticated *dynamic*.

Static

One approach to mining over continuous data is to transform quantitative attributes into discrete, pre-determined nominal attributes in advance.

Example:

For the domain *Age* = (*0..120*), a value such as "25" would be transformed to the item *Age=18-25* and a value "99" would be transformed to the item *Age=75 and over*.

The pre-determined method is generally most appropriate where the intervals have meaning in the domain, generally related to where consequent actions can be targeted. For example, partitioning *age* into the year groups that correspond to educational stages might be helpful for targeting education-related interventions.

This transformation need not influence the data mining algorithm, that is, a nominal-only algorithm like *apriori* can then be used effectively.

Dynamic

A more sophisticated approach is to cluster attribute values into bins relying on the data distribution, and then further combining these bins during the mining process. In this case we are looking for a dynamic discretisation that interacts with the mining algorithm to choose good value ranges that, for example, maximise the confidence of rules. That is, you need to choose a mining method that is aware of the continuous values and an off-the-shelf apriori algorithm will not do.

One approach to this problem is to integrate the frequent item-set discovery with a data warehouse cube structure, so that the aggregate counts over multiple dimensions in the cube can be retrieved in a top-down fashion, (either drilling down over concept hierarchies or drilling down to introduce additional dimensions) and stopping the drill-down when a count fails minimum support (because the apriori condition ensures that no lower cube will satisfy minimum support) and using the dimensional labels at that level as the bins for nominal items.

An alternative typical approach is to integrate a *clustering* method (covered later in the course). For each quantitative attribute a clustering algorithm is applied to each dimension to satisfy minimum support. Then for each such cluster, we look to combine it with another dimension to find a 2-D cluster with minimum support, and so on to higher dimensions. The range of attribute values that occur in a cluster then define the discrete bins for nominal items. The apriori condition tells us that we need not look to combine a cluster with another once it fails to satisfy minimum support.

Interpretation of rules over transformed data

If the data has been transformed as discussed here, then it is important to be aware of this when interpreting the rules and the evaluation metrics for the rules. For example, rules that contain no *Disease* item may be considered irrelevant and so uninteresting, irrespective of the interestingness measures. For another example, when some quantitative variable say *age*, has been split into two intervals by the median value, and so every transaction has either one interval value or the other, both items will be highly frequent with 50% support. Assuming the other items are more scattered, pretty much *every* frequent itemset will include one *age* or the other, but never both, so you should be very careful in concluding that the presence of one in a frequent itemset is somehow significant.

6.3. Advanced applications (Text:7.6.2)

Frequent pattern mining, despite being a rather simple idea, has proven to be very successful. Its success may be due partly to its ability to yield readable rules that can make sense for application. Other than market basket analysis, there are many other application areas.

Pattern mining is widely used for **noise filtering and data cleaning**. For example, microarray data for studying gene expression has tens of thousands of dimensions (this is often called the " $p \gg n$ " problem) and it can be very noisy, so pattern mining offers a useful pre-processing tool because items that occur frequently together are likely to be significant rather than noise. We can then focus our analysis on those variables for which some values occur in rules with high support.

Pattern mining is useful for **data exploration** -- to discover inherent structures and clusters hidden in the data. These structures can then be used as basic building blocks for classification problems.

In high dimensional space, frequent patterns can be used for **clustering**, that is to define lower-dimensional sub-space features based on common frequent patterns.

Pattern mining is used in **spatio-temporal data**, time series data, **image data**, **video data** and **multimedia** data for identifying patterns of co-location, and identifying frequently occurring image fragments as the basis for clustering and description.

Pattern mining has been used for analysis of **sequence or structural data** such as trees and graphs. For example, there have been successes in discovering both plagiarism and copy-paste bugs in large software programs.

Pattern mining has been known to help in designing simple indexing structures for large complex data sets to support **similarity search** in XML document databases and chemical compound databases.

Recommender systems have employed pattern mining for discovery of patterns in customer behaviour.

Often these applications require specialist techniques to handle the particular quirks of the data and the problem domain.

7. Practical Exercises

Here are shortcuts to all of the practical exercises within this book.

Exercise 1: <https://wattlecourses.anu.edu.au/mod/book/view.php?id=2172374&chapterid=402620>

Exercise 2: <https://wattlecourses.anu.edu.au/mod/book/view.php?id=2172374&chapterid=402617>

8. Quiz

ACTION: **blended only** Please take the weekly quiz now.

ACTION: **on campus only** Please take the weekly quiz in your lab class.



Week 4 quiz: Association mining