***** Advanced Machine Learning *****
Association Rule Learning

M. Sc. Daniel Wehner

SAP SE / DHBW Mannheim

Summer term 2020

DHBW
Duale Hochschule
Baden-Württemberg

Agenda for this Unit

Introduction

What is Association Rule Mining?

- Association rule mining belongs to the category of unsupervised learning.

- Association rules describe frequent co-occurrences in the data (**not necessarily causality!**)

- Examples:

  - Market basket analysis (*Which products are frequently bought together? E. g. Amazon*)

  - Course schedule planning (*Which courses are often attended together?*)

  - Other use cases: Marketing promotions, inventory management, customer relationship management (CRM)

- The general form of a rule is given by:

$$\overbrace{\{a_1, a_2, \ldots, a_n\}}^{\textbf{Antecedent}} \rightarrow \overbrace{\{b_1, b_2, \ldots, b_m\}}^{\textbf{Consequent}} \tag{1}$$

- Example: $\{bread, cheese\} \rightarrow \{wine\}$

Figure 1: Famous example from Amazon

Important Terminology

- Suppose $\mathcal{I}$ is a set of unique items which we have in our portfolio, $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ is a list of transactions (what customers bought).

- Each transaction $t_i \in \mathcal{T}$ is an element of $\mathfrak{P}(\mathcal{I})$, the power set of $\mathcal{I}$. (**What is a power set?**)

- Example:

Id	Transactions
1	$\{beer, chips, wine\}$
2	$\{beer, chips\}$
3	$\{pizza, wine\}$
4	$\{chips, pizza\}$

Id	beer	chips	pizza	wine
1	1	1	0	1
2	1	1	0	0
3	0	0	1	1
4	0	1	1	0

| Figure 2: | Left: List of transactions (raw), right: List of transactions in binary form |

⚠ **Simplification: We ignore quantities and prices of the items sold.**

Item sets

- A collection of k items is called k-item set.

- Example: $\{pizza, wine\}$ is a 2-item set.

- The number of items contained in a transaction t_i is sometimes referred to as the **transaction width** $w(t_i) = |t_i|$.

- An important property of an item set X is the **support count σ**:

$$\sigma(X) = |\{t_i | X \subseteq t_i \wedge t_i \in \mathcal{T}\}| \tag{2}$$

- **What does the support count tell us?** $\sigma(X)$ refers to the number of transactions X occurs in.

Quality Measures

- *Question:* How to measure the quality of an association rule?

- **Support:**

  - Proportion of examples for which head and body are true.

  - Example $A \rightarrow B$: How many customers bought A and B together?

$$\text{support}(A \rightarrow B) = \text{support}(A \cup B) = \frac{\sigma(A \cup B)}{n} \tag{3}$$

- **Confidence:**

  - Proportion of examples for which the head is true among those for which the body is true.

  - Example: If customers bought A, how likely are they to also buy B?

$$\text{confidence}(A \rightarrow B) = \frac{\text{support}(A \cup B)}{\text{support}(A)} = \frac{\sigma(A \cup B)}{\sigma(A)} \tag{4}$$

- Support: There is a huge number of possible rules, but not all of them are interesting.
  ⇒ **Prune (remove) rules with low support.**

- Confidence: The higher the confidence the more reliable is the rule.

- Example:

  – $R = \{bread, cheese\} \rightarrow \{wine\}$

  – support$(R) = 0.01$ and confidence$(R) = 0.8$

  – 80 % of all customers who bought bread and cheese also bought red wine.

  – However, only 1 % of the customers bought all three items together.

Apriori

Learning Problem

- The **Apriori algorithm** can be used to find association rules.

- The learning problem can be summarized as follows:
  Given a set of transactions $\mathcal{T}$, find all rules having support $\geqslant s_{min}$ and confidence $\geqslant c_{min}$, where s_{min} and c_{min} are thresholds.

- Obviously, mining all possible rules is super expensive.

$$|\text{rules}| = 3^d - 2^{d+1} + 1 \qquad \text{where} \qquad d \equiv |\mathcal{T}| \tag{5}$$

- Also, rules can be spurious (i. e. patterns may occur by chance and are not systematic).

⚠ **We have to avoid considering all possible rules! $\Rightarrow$ Employ early pruning.**

Early Pruning

- The goal is to generate rules which have high support and high confidence.

- Observation: If an item set is infrequent (does not have sufficient support), calculating the confidence can be omitted.

- As a consequence, all rules which can be generated from this item set do not have to be considered anymore.

- Example for the item set $A = \{beer, diapers, milk\}$:

  – The rules derived from item set A are given below.

  – If we know item set A to be infrequent, we can prune all these rules.

  – There is no need to calculate the confidence for these rules (**decoupling of support and confidence**).

$$\{beer, diapers\} \rightarrow \{milk\} \qquad\qquad \{beer, milk\} \rightarrow \{diapers\}$$
$$\{diapers, milk\} \rightarrow \{beer\} \qquad\qquad \{beer\} \rightarrow \{diapers, milk\}$$
$$\{milk\} \rightarrow \{beer, diapers\} \qquad\qquad \{diapers\} \rightarrow \{beer, milk\}$$

Apriori Algorithm

- The overall algorithm consists of two major steps:

  1. **Frequent item set generation:**
     Find all item sets which have sufficient support (satisfy the support constraint).

  2. **Rule generation:**
     Extract highly confident rules which satisfy the confidence constraint.

- In the following we will have a closer look at these two steps.
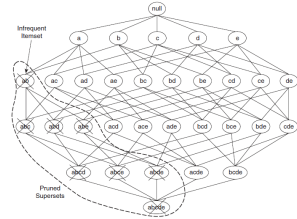
Step 1) Frequent item set generation

- It is possible to enumerate all possible item sets with a lattice $\Rightarrow$ fig. 3.

- A brute force approach could calculate the support for each candidate set and rank them by the result.

- **Problem:** The number of candidate sets grows exponentially with $|\mathcal{I}|$: $2^{|\mathcal{I}|} - 1$ (excluding empty set).

- Example: For $\mathcal{I} = \{a, b, c, d, e\}$, we have 31 possible candidates.

- Therefore, the candidate sets should be generated more efficiently.

- We can make use of the **anti-monotonicity** of the support:
   If an item set is frequent, then all of its subsets must be frequent as well. Also, if an item set is infrequent, then all its supersets must be infrequent too.

- Adding a condition can never increase the support of a rule:

$$A \subseteq B \Longrightarrow \text{support}(A) \geqslant \text{support}(B) \tag{6}$$

- **An item set can only be frequent, if all its subsets are frequent and all supersets of an infrequent item set are also infrequent.**
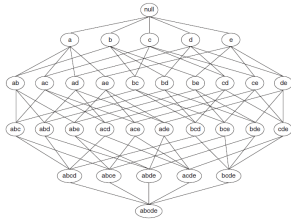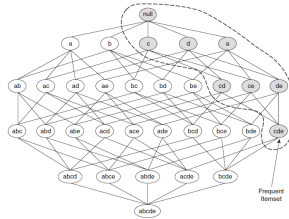
(a) item set lattice   (b) frequent item set   (c) infrequent item set

Figure 3:   Item set lattice for $\mathcal{I} = \{a, b, c, d, e\}$

1. $k \leftarrow 1$

2. $C_1 \leftarrow \mathcal{I}$

3. **while** $C_k \neq \emptyset$ **do**

   ▷ $S_k \leftarrow C_k \backslash \{$all infrequent item sets in $C_k\}$

   ▷ $C_{k+1} \leftarrow$ all sets with $k+1$ elements which can be formed by uniting two item sets in S_k

   ▷ $C_{k+1} \leftarrow C_{k+1} \backslash \{$item sets, where not all subsets of size k are in $S_k\}$

   ▷ $S \leftarrow S \cup S_k$

   ▷ $k \leftarrow k + 1$

4. **return** S

⚠ The algorithm leaves it open how the candidate set C_{k+1} is generated. How can this be done efficiently?

- Requirements for efficient candidate generation:

  - We have to avoid producing too many candidates.

  - At the same time we have to ensure that all frequent item sets are found (**completeness**)

  - We don't want to produce duplicates (**efficiency**)

- The Apriori algorithm uses the following method:

  - Merge a pair of k-item sets only if their first $k - 1$ items are identical.

$$A = \{a_1, a_2, \ldots, a_k\} \qquad\qquad B = \{b_1, b_2, \ldots, b_k\} \qquad\qquad (7)$$

  - Merge A and B, if $a_j = b_j \ (j = 1, 2, \ldots, k - 1) \wedge a_k \neq b_k$

  - Example:

    ▷ $A = \{bread, milk, pizza\}, B = \{bread, milk, wine\}$

    ▷ A and B are merged into $\{bread, milk, pizza, wine\}$.

  - This method still requires pruning non-frequent item sets.

  - **Important: The item sets have to be in lexicographic order.**

Let's calculate the frequent item sets from the introductory example ($s_{min} = 0.25$):

Id	beer	chips	pizza	wine
1	1	1	0	1
2	1	1	0	0
3	0	0	1	1
4	0	1	1	0

$$C_1 = \{\{beer\}, \{chips\}, \{pizza\}, \{wine\}\}$$
$$S_1 = \{\{beer\}, \{chips\}, \{pizza\}, \{wine\}\}$$
$$C_2 = \{\{beer, chips\}, \{beer, pizza\}, \{beer, wine\}, \{chips, pizza\}, \{chips, wine\}, \{pizza, wine\}\}$$
$$S_2 = \{\{beer, chips\}, \{beer, wine\}, \{chips, pizza\}, \{chips, wine\}, \{pizza, wine\}\}$$
$$C_3 = \{\{beer, chips, wine\}, \{chips, pizza, wine\}\}$$
$$S_3 = \{\{beer, chips, wine\}\}$$
$$C_4 = \emptyset$$
$$S = \bigcup_{k=1}^{3} S_k$$

- The search space for frequent item sets can be structured using the subset relationship.

- **Border:**

  - The border $\Rightarrow$ fig. 4 consists of all item sets for which...

    ▷ ...all subsets are frequent and...

    ▷ ...no superset is frequent.

  - **Positive border:** Elements of the border which are frequent.

  - **Negative border:** Elements of the border which are infrequent.

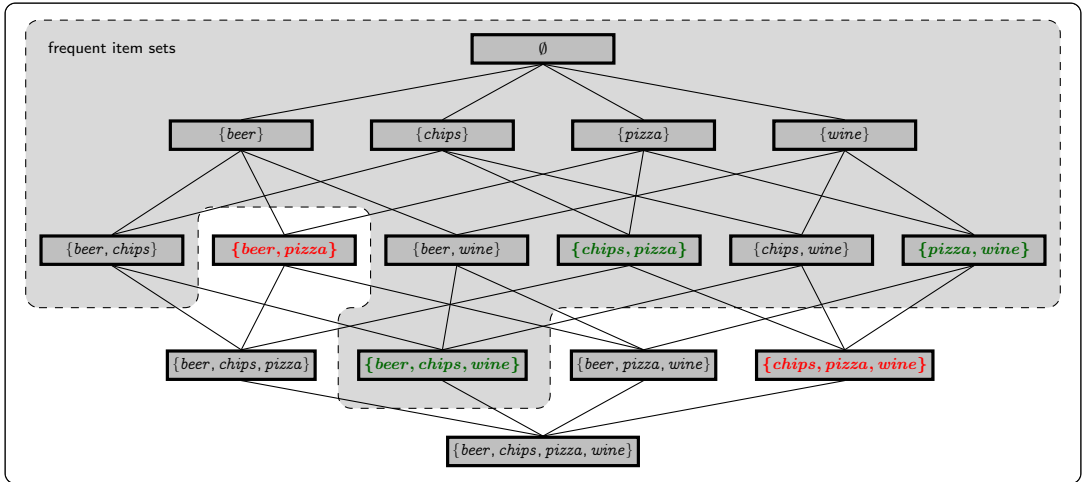⚠ **Frequent item sets = positive border plus all subsets of border elements**

Figure 4: Border for the example above
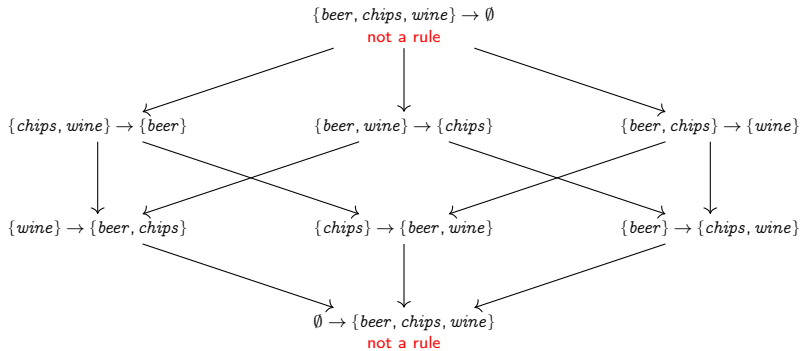
Step 2) Generation of association rules

- The frequent item sets can now be used to generate association rules.

- For each frequent k-item set X, there are $2^k - 2$ possible association rules (without $X \rightarrow \emptyset$ and $\emptyset \rightarrow X$) of the general form $\Rightarrow$ fig. 5:
$$A \rightarrow B \qquad \text{with} \qquad A \cup B = X \wedge A \cap B = \emptyset \tag{8}$$

- Calculate the confidence for the rules and check whether they fulfill the confidence constraint.

- We can also define **anti-monotonicity** for the confidence:
  If a rule is not confident, moving conditions from body to head results in rules which are also not confident.

$$\text{confidence}(A \rightarrow B \cup C) \leqslant \text{confidence}(A \cup B \rightarrow C) \tag{9}$$

- This circumstance can again be used for pruning the search space!

$\{beer, chips, wine\} \rightarrow \emptyset$
not a rule

$\{chips, wine\} \rightarrow \{beer\}$ $\{beer, wine\} \rightarrow \{chips\}$ $\{beer, chips\} \rightarrow \{wine\}$

$\{wine\} \rightarrow \{beer, chips\}$ $\{chips\} \rightarrow \{beer, wine\}$ $\{beer\} \rightarrow \{chips, wine\}$

$\emptyset \rightarrow \{beer, chips, wine\}$
not a rule

Figure 5:   Search space for association rules (frequent item set $\{beer, chips, wine\}$)

Let's make a full example for the Apriori algorithm ($s_{min} = 0.5$, $c_{min} = 1.0$):

Id	bread	butter	coffee	milk	sugar
1	1	1	0	0	1
2	0	0	1	1	1
3	1	0	1	1	1
4	0	0	1	1	0

$$C_1 = \{\{bread\}, \{butter\}, \{coffee\}, \{milk\}, \{sugar\}\}$$

$$S_1 = \{\{bread\}, \{coffee\}, \{milk\}, \{sugar\}\}$$

$$C_2 = \{\{bread, coffee\}, \{bread, milk\}, \{bread, sugar\}, \{coffee, milk\}, \{coffee, sugar\}, \{milk, sugar\}\}$$

$$S_2 = \{\{bread, sugar\}, \{coffee, milk\}, \{coffee, sugar\}, \{milk, sugar\}\}$$

$$C_3 = \{\{coffee, milk, sugar\}\}$$

$$S_3 = \{\{coffee, milk, sugar\}\}$$

$$C_4 = \emptyset$$

$$S = \bigcup_{k=1}^{3} S_k$$

- Rules with $c_{min} = 1.0$:

$\{bread\} \rightarrow \{sugar\}$	$s = 0.50$	$c = 1.00$
$\{milk\} \rightarrow \{coffee\}$	$s = 0.75$	$c = 1.00$
$\{coffee\} \rightarrow \{milk\}$	$s = 0.75$	$c = 1.00$
$\{milk, sugar\} \rightarrow \{coffee\}$	$s = 0.50$	$c = 1.00$
$\{coffee, sugar\} \rightarrow \{milk\}$	$s = 0.50$	$c = 1.00$

- Other rules are either not frequent enough and are filtered out in step 1;
  e. g. $\{butter\} \rightarrow \{bread, sugar\}$, for which $s = 0.25$ and $c = 1.0$...

- ...or not confident enough and filtered out in step 2;
  e. g. $\{milk, coffee\} \rightarrow \{sugar\}$, for which $s = 0.5$ and $c = 0.67$.

Miscellaneous

Interestingness

- **Problem:** There might still be way too many rules.

- Assume the following two rules:

$$R_1 = A \cup B \to C \qquad\qquad R_2 = A \to C \qquad\qquad (10)$$

- Filter out R_1, if the rule is...

  - ...*trivial* (R_2 covers the same examples)

  - ...*unproductive* (R_2 has equal or higher confidence)

  - ...*insignificant* (Confidence of R_2 is not significantly worse)

- Filter by **interestingness** (*How can we measure interestingness?*)

- Support and confidence are not sufficient to capture whether a rule is interesting or not.

- A rule may have high support and confidence, but still may not be interesting.

- Example:

  - Consider the rule: $\{diapers\} \rightarrow \{beer\}$; $c = 0.90$

  - 90 % of all customers who buy diapers also buy beer.

  - Sounds like and interesting association rule.

  - **But:** If we know, that 90 % of all customers buy beer, this rule is not interesting anymore.

Lift, Leverage and Conviction

- Consider rule $R = A \rightarrow B$

- **Lift:** Rule R is interesting, if lift$(R) \gg 1$.

$$\text{lift}(A \rightarrow B) = \frac{\text{support}(A \rightarrow B)}{\text{support}(A) \cdot \text{support}(B)} \tag{11}$$

- **Leverage:** Rule R is interesting, if leverage$(R) \gg 0$.

$$\text{leverage}(A \rightarrow B) = \text{support}(A \rightarrow B) - \text{support}(A) \cdot \text{support}(B) \tag{12}$$

- **Conviction:** Expected ratio that A occurs without B (incorrect prediction of R).

$$\text{conviction}(A \rightarrow B) = \frac{1 - \text{support}(B)}{1 - \text{confidence}(A \rightarrow B)} \tag{13}$$

Thank you very much for the attention!

Topic: ***** Advanced Machine Learning ***** Association Rule Learning
Term: Summer term 2020

Contact:
M. Sc. Daniel Wehner
SAP SE / DHBW Mannheim
daniel.wehner@sap.com

Do you have any questions?