

# Association rule mining and recommendation systems

"If a customer buys potato chips and sour cream, he/she will buy dip sauce"

# Learning goals

1. Understand the principles of association analysis.
2. Understand the principles of recommendation systems.
3. Learn to apply a recommendation algorithm in Python.

# Association and recommendation systems

- Association rule mining
  - Find rules for items that occur together in a transaction more often than expected.
- Recommendation systems
  - Find items that other users have ranked highly.

# Association analysis

- Association analysis (or association rule mining) is one of the central data mining methods.
- The most well-known application is shopping basket analysis (enables targeted marketing, optimal placement of products etc.).
  - Also applications in medical genetics, website analytics, etc..
- Data is seen as a set of *transactions*.
- Each transaction contains a set of *items*.
- The goal is to find items that often co-exist (in same transactions).
- These sets of items, or patterns, are expressed as association rules.
  - Each rule has a direction.

# Association rule

- "If a customer buys pea soup, he/she will buy mustard."
- "If a customer buys potato chips and sourcream, he/she will buy dip sauce"

Or more formally:

- {Pea soup}  $\rightarrow$  {Mustard}
- {Potato chips, Sourcream}  $\rightarrow$  {Dip sauce}

# Example: shopping basket analysis (transaction encoding)

ID	Items
1	HYG, DAI, SWE, MEA
2	DAI, SWE; MEA
3	FRU, DAI, SWE
4	FRU, HYG, DAI, MEA
5	HYG, DAI, SWE
6	HYG, DAI, MEA
7	HYG, MEA

- Seven transactions.
- Items:
  - **FRU**: fruit and vegetables
  - **HYG**: hygiene products
  - **DAI**: dairy products
  - **SWE**: sweets
  - **MEA**: meat and poultry

## Example: shopping basket analysis (table encoding)

ID	FRU	HYG	DAI	SWE	MEA
1	0	1	1	1	1
2	0	0	1	1	1
3	1	0	1	1	0
4	1	1	1	0	1
5	0	1	1	1	0
6	0	1	1	0	1
7	0	1	0	0	1

- Alternatively, presence of items in a transaction can be coded by dummy variables.
- The Python implementation (apyori) uses transaction encoding.
  - Reformatting needed as a preprocessing step.

# Association rules

ID	Items
1	HYG, DAI, SWE, MEA
2	DAI, SWE; MEA
3	FRU, DAI, SWE
4	FRU, HYG, DAI, MEA
5	HYG, DAI, SWE
6	HYG, DAI, MEA
7	HYG, MEA

- We can generate the following association rules, for example:
  - $\{DAI\} \rightarrow \{SWE\}$
  - $\{SWE\} \rightarrow \{DAI\}$
  - $\{MEA\} \rightarrow \{FRU\}$
  - $\{MEA, DAI\} \rightarrow \{SWE\}$
- Some rules are 'better' than others.
- How to measure the goodness of the rules?



# Support

- The support of an association rule means the number of the transactions with all items in the rule divided by the total number of transactions.
- High support means that the rule can be applied often.
- Low support may indicate that the rule just represent random occurrences.

ID	Items
1	HYG, DAI, SWE, MEA
2	DAI, SWE; MEA
3	FRU, DAI, SWE
4	FRU, HYG, DAI, MEA
5	HYG, DAI, SWE
6	HYG, DAI, MEA
7	HYG, MEA

The support of rule **{MEA, DAI} → {SWE}** is  $2/7 \approx 0.29$ , since the items in set **{MEA, DAI, SWE}** are present in two transactions. The total number of transactions is 7.

# Confidence

- Confidence tells how probable it is to see the items of the right side given that the items of the left side are present in the transaction.
- High confidence means that the rule is 'true', or reliable, often.

ID	Items
1	HYG, DAI, SWE, MEA
2	DAI, SWE; MEA
3	FRU, DAI, SWE
4	FRU, HYG, DAI, MEA
5	HYG, DAI, SWE
6	HYG, DAI, MEA
7	HYG, MEA

The confidence of rule **{MEA, DAI} → {SWE}** is  $2/4 = 0.50$ , since the items in set **{MEA, DAI, SWE}** are present in two transactions whereas the items of set **{MEA, DAI}** are present in four transactions.

# Example

- What are the support and confidence of the following association rules?

ID	Items
1	HYG, DAI, SWE, MEA
2	DAI, SWE; MEA
3	FRU, DAI, SWE
4	FRU, HYG, DAI, MEA
5	HYG, DAI, SWE
6	HYG, DAI, MEA
7	HYG, MEA

- **{DAI} → {SWE}**
- **{SWE} → {DAI}**
- **{FRU} → {HYG, MEA}**
- **{FRU, SWE} → {DAI}**
- **{FRU} → {SWE, MEA}**

# Goal

Find association rules with high enough support and high enough confidence.

- Challenges
  - Algorithmic complexity
  - Definition of parameters by trial and error
    - Lower bound for support and confidence.
  - Interpretation of the result (real vs random)
    - Problem if costly decisions are based on the results.
  - How to deal with continuous variables (need to discretize)?
- In the following example we take a look at automated search of association rules (i.e. *association rule mining*).

# Apriori algorithm

- It is inefficient to look for association rules by testing all possible rules.
- Apriori algorithm (*Agrawal et. al., 1994* ) is a classical search algorithm for association rules.
  - In Python, implemented as **apriori()** function in **apriori** library
- Main idea:
  - First find all frequent item sets of size 1 (enough support).
  - Then those of size 2, size 3, etc.
  - When all the frequent item sets have been discovered, build all possible association rules out of them.
  - From this set, pick the rules with high enough confidence.

# Example

ID	Items
1	HYG, DAI, SWE, MEA
2	DAI, SWE; MEA
3	FRU, DAI, SWE
4	FRU, HYG, DAI, MEA
5	HYG, DAI, SWE
6	HYG, DAI, MEA
7	HYG, MEA

- Let's set the minimum support at  $3/7$  ( $= 0.429$ ).
  - Thus, we first search for item sets that are present at least 3 times.
- Let's set the minimum confidence at 0.75.
- We start by calculating the frequencies of all single-item sets.

## Example: size 1 candidate sets

- C1, size one candidate sets

Item set	Frequency
FRU	2
HYG	5
DAI	6
SWE	4
MEA	5

- The frequency of each item is displayed.
- Next, prune those items whose frequency falls below the minimum support (3).

## Example: size 1 frequent sets

• L1, size 1 frequent sets	
Item set	Frequency
HYG	5
DAI	6
SWE	4
MEA	5

- Now the frequency of each item is at least that dictated by the minimum support.
- Next, combine the items into sets of size 2.



## Example: size 2 candidate sets

• C2, size 2 candidate sets	
Item set	Frequency
HYG, DAI	4
HYG, SWE	2
HYG, MEA	4
DAI, SWE	4
DAI, MEA	4
SWE, MEA	2

- There will be  $3+2+1 = 6$  combinations.
- The frequency of each set is computed.
- Sets whose frequency is below the threshold, will be pruned.

## Example: size 2 frequent sets

• L2, size 2 frequent sets	
Item set	Frequency
HYG, DAI	4
HYG, MEA	4
DAI, SWE	4
DAI, MEA	4

- Now two-items sets with a high enough frequency remain.
- Next, find three-item sets that are obtained by combining two-item sets.

## Example: size 3 candidate sets

- C3, size 3 candidate sets

Item set	Accepted?
HYG, DAI, MEA	Accepted.
HYG, DAI, SWE	Not accepted, since HYG, SWE is not in L2.
DAI, SWE, MEA	Not accepted, since SWE, MEA is not in L2.
HYG, MEA, SWE	Not accepted, since SWE, MEA is not in L2.

- For each three-items set check that all its proper subsets of size 2 belong to set L2.
- This is due to the fact that a set of size  $n$  can't be frequent if it has a non-frequent subset of size  $n-1$ .

## Example: size 3 candidate sets

•  
C3

Item set	Frequency
HYG, DAI, MEA	3

- One set of size 3 remains.
- Let's check whether its frequency is high enough.

## Example: size 3 frequent sets

• L3, size 3 frequent sets	
Item set	Frequency
HYG, DAI, MEA	3

- The only remaining set is frequent enough.
- Sets of size 4 can't be formed.
- Next, let's summarize the discovered frequent sets.

## Example: frequent sets

- L1, L2 and L3 combined

Item set	Frequency
HYG	5
DAI	6
SWE	4
MEA	5
HYG, DAI	4
HYG, MEA	4
DAI, SWE	4
DAI, MEA	4
HYG, DAI, MEA	3

- Next, we deduct association rules from sets with at least two items.
- Each subset is taken in turn as the left side of the rule.
- The remaining item(s) form the right side.

## Example: computation of association rules

Item set	Association rule	Confidence
HYG, DAI	$\{HYG\} \rightarrow \{DAI\}$	4/5
	$\{DAI\} \rightarrow \{HYG\}$	4/6
HYG, MEA	$\{HYG\} \rightarrow \{MEA\}$	4/5
	$\{MEA\} \rightarrow \{HYG\}$	4/5
DAI, SWE	$\{DAI\} \rightarrow \{SWE\}$	4/6
	$\{SWE\} \rightarrow \{DAI\}$	4/4
DAI, MEA	$\{DAI\} \rightarrow \{MEA\}$	4/6
	$\{MEA\} \rightarrow \{DAI\}$	4/5
HYG, DAI, MEA	$\{HYG, DAI\} \rightarrow \{MEA\}$	3/4
	$\{HYG, MEA\} \rightarrow \{DAI\}$	3/4
	$\{DAI, MEA\} \rightarrow \{HYG\}$	3/4
	$\{HYG\} \rightarrow \{DAI, MEA\}$	3/5
	$\{DAI \rightarrow HYG, MEA\}$	3/6
	$\{MEA \rightarrow HYG\}, \{DAI\}$	3/5

- Finally, we pick the rules whose support is equal to or higher than the threshold (0.75).

## Example: final set of association rules

Association rule	Support	Confidence
$\{HYG\} \rightarrow \{DAI\}$	5/7	4/5
$\{HYG\} \rightarrow \{MEA\}$	5/7	4/5
$\{MEA\} \rightarrow \{HYG\}$	5/7	4/5
$\{SWE\} \rightarrow \{DAI\}$	4/7	4/4
$\{MEA\} \rightarrow \{DAI\}$	5/7	4/5
$\{HYG, DAI\} \rightarrow \{MEA\}$	4/7	3/4
$\{HYG, MEA\} \rightarrow \{DAI\}$	4/7	3/4
$\{DAI, MEA\} \rightarrow \{HYG\}$	4/7	3/4



# Apriori in Python

- The **scikit-learn** library used on the course does not contain association rule mining functionality.
- For an example of a market basket analysis with **apyori** package, see **Methods/Data/Market Basket (demo)** on the **Documents** tab in the course's Oma workspace.
  - The package needs to be installed in the Anaconda prompt:  
**pip install apyori.**

# Alternatives

- Apriori algorithm can be further optimized.
  - Usage of more advanced data structures.
  - Reduction of the need to scan the data more than necessary.
- Other algorithms
  - FPGrowth
    - Avoids the generation of large candidate sets (cf. Apriori).
  - Eclat
    - Each item set is expanded until the support threshold is reached (cf. Apriori where first all sets of size 2 are created, then all sets of size 3, etc.).

# Important aspects

- The first shot of association analysis seldom produces a satisfactory result.
  - It is hard to find a suitable combination of parameter values beforehand.
  - In practice, one has to play with parameter values and find such a set of values that produce some results.
  - As a consequence, a result is always obtained.
  - How can I know whether it is statistically significant?
    - Significance can be evaluated empirically by randomization tests.
- The method is designed to work with discrete variables.
  - Continuous variables must first be discretized.
  - Finding the optimal discretization is alone a hard problem (NP complete).


# Recommendation systems

- Association rule mining can be applied for recommendations.
- Recommendation systems (aka. recommendation engines) make recommendations based on user behaviour and/or rankings
- Approaches
  - User-based
    - Find similar users based on rankings. Recommend items ranked high by similar users.
  - Item-based
    - Find item sets that are ranked in a similar fashion by users.
  - Keyword-based

# kNN in collaborative filtering

- K nearest neighbours (KNN) is a simple method for finding similar observations.
  - Place users (or items) in the multidimensional feature space based on their ratings.
  - When making the recommendations, find the  $k$  nearest neighbours in the feature space.
  - Base recommendation on the neighbours' choices.

# Example of kNN-based recommendation

Rec. order				2	1		kNN recommendation with cosine similarity (ignore missing data)										 <div>k</div> <div>2</div>		
Rating rank (unseen)				4	1														
User	5	1	2			Avg 2,666667	1	1	1	0	0								
	M1	M2	M3	M4	M5	Avg	M1	M2	M3	M4	M5	Len.	Sc. prod.	Prod. len.	cossim	Rank	Neighbor		
Mary	4	1	1	1	5	2,4	1	1	1	0	0	4,24	23	23,24	0,990	2	1		
Peter	1	1	5	4	2	2,6	1	1	1	0	0	5,20	16	28,46	0,562	6	0		
Ahmed	4	2	1	5	2	2,8	1	1	1	0	0	4,58	24	25,10	0,956	3	0		
Olga	1	5	5	5		4	1	1	1	0	0	7,14	20	39,12	0,511	7	0		
Mika	5	1	2	1	5	2,8	1	1	1	0	0	5,48	30	30,00	1,000	1	1		
Anni	4	5	1		1	2,75	1	1	1	0	0	6,48	27	35,50	0,761	5	0		
Tim		2	2	4	1	2,25	0	1	1	0	0	2,83	6	6,32	0,949	4	0		
Relevant rating rank	2	4	3	4	1														
Mean relevant rating	4,5	1	1,5	1	5														
	M1	M2	M3	M4	M5														
	4	1	1	1	5														
	5	1	2	1	5														

Scalar product of usable ratings vectors

Product of lengths of usable ratings vectors

indicators of usable ratings

length of usable ratings vector

cos(θ) =  $\frac{A \cdot B}{\|A\| \|B\|}$

indicator for belonging to nearest neighbors

ratings of relevant users

# Recommendation systems in Python

- Again, **scikit-learn** doesn't provide recommendation engine functionality.
- A package called **scikit-surprise** can be used instead.
  - <http://surpriselib.com/>
- For an example, see **Methods/Data/Series (demo)** on the **Documents** tab in the course's Oma workspace.