Foundations of Knowledge Management: Association Rules

(LV: 707.009)

12.01.2010

Mark Kröll
mkroell@tugraz.at
Graz University of Technology, Austria
Today’s Outline

- Association Rules
  - Motivating Example
  - Definitions
  - The Apriori Algorithm
  - Limitations / Improvements

Acknowledgements / slides based on:
- Lecture „Data Management and Exploration“ by Thomas Seidl (RWTH Aachen)
- Lecture “Association Rules” by Berlin Chen
- Lecture “PG 402 Wissensmanagment” by Z. Jerroudi
- Lecture “LS 8 Informatik Computergestützte Statistik“ by Morik and Weihs
- Association Rules by Prof. Tom Fomby
Today we learn

- Why Association Rules are useful?
  - history + motivation
- What Association Rules are?
  - definitions
- How we can mine them?
  - the Apriori algorithm
  - Illustrating example
- Which challenges they face?
  - + means to address them
Why do we need association rule mining at all?
Motivation for Association Rules (1)

Association Rule Mining can help to better understand purchase behavior!

For instance, \{beer\} => \{chips\}
Market Basket Analysis (MBA)(1)

- In retailing, *most purchases are bought on impulse*. Market basket analysis gives clues as to what a customer might have bought *if the idea had occurred to them*.
  - decide the location and promotion of goods inside a store.

Observation: Purchasers of Barbie dolls are more likely to buy candy.
  \{barbie doll\} => \{candy\}
  - place high-margin candy near to the Barbie doll display.

Create Temptation: Customers who would have bought candy with their Barbie dolls *had they thought of it will now be suitably tempted.*
Market Basket Analysis (MBA)(2)

- Further possibilities:
  - comparing results between different stores, between customers in different demographic groups, between different days of the week, different seasons of the year, etc.
  - If we observe that a rule holds in one store, but not in any other then we know that there is something interesting about that store.
    - different clientele
    - different organization of its displays (in a more lucrative way …)

→ investigating such differences may yield useful insights which will improve company sales.
ReCap: Let’s go shopping

- **Objective of Association Rule Mining:**
  - find **associations** and **correlations** between different items (products) that customers place in their shopping basket.
  - to better predict, e.g.,:
    - (i) what my customers buy?  
    - (ii) when they buy it?  
    - (iii) which products are bought together?
      - (→ spectrum of products)
      - (→ advertising)
      - (→ placement)
Introduction into AR

- **Formalizing the problem a little bit**
  - Transaction Database $T$: a set of transactions $T = \{t_1, t_2, \ldots, t_n\}$
  - Each transaction contains a set of items (item set)
  - An item set is a collection of items $I = \{i_1, i_2, \ldots, i_m\}$

- **General Aim:**
  - Find frequent/interesting patterns, associations, correlations, or causal structures among sets of items or elements in databases or other information repositories.
  - Put this relationships in terms of association rules
    
    \[ X \Rightarrow Y \]
    
    - where $X$, $Y$ represent two itemsets
Examples of AR

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>bread, jelly, peanut-butter</td>
</tr>
<tr>
<td>T2</td>
<td>bread, peanut-butter</td>
</tr>
<tr>
<td>T3</td>
<td>bread, milk, peanut-butter</td>
</tr>
<tr>
<td>T4</td>
<td>beer, bread</td>
</tr>
<tr>
<td>T5</td>
<td>beer, milk</td>
</tr>
</tbody>
</table>

- **Frequent Item Sets:**
  - Items that appear frequently together
    - \( I = \{\text{bread, peanut-butter}\} \)
    - \( I = \{\text{beer, bread}\} \)

**Examples:**

- **Quality?**
- \( \text{bread} \Rightarrow \text{peanut-butter} \)
- \( \text{beer} \Rightarrow \text{bread} \)

**Reads as:**
If you buy bread, then you will peanut-butter as well.
What is an interesting rule?

- **Support Count (σ)**
  - Frequency of occurrence of an itemset
    - \( \sigma (\{\text{bread, peanut-butter}\}) = 3 \)
    - \( \sigma (\{\text{beer, bread}\}) = 1 \)

- **Support (s)**
  - Fraction of transactions that contain an itemset
    - \( s(\{\text{bread, peanut-butter}\}) = 3/5 \ (0.6) \)
    - \( s (\{\text{beer, bread}\}) = 1/5 \ (0.2) \)

- **Frequent Itemset**
  - = an itemset whose support is greater than or equal to a minimum support threshold (\( \text{minsup} \))
What is an interesting rule?

- An association rule is an implication of two itemsets
  \[ X \Rightarrow Y \]

- Most common measures:
  - **Support** (\( s \))
    - The occurring frequency of the rule, i.e., the number of transactions that contain both \( X \) and \( Y \)
  - **Confidence** (\( c \))
    - The strength of the association, i.e., measures the number of how often items in \( Y \) appear in transactions that contain \( X \) vs. the number of how often items in \( X \) occur in general

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>bread, jelly, peanut-butter</td>
</tr>
<tr>
<td>T2</td>
<td>bread, peanut-butter</td>
</tr>
<tr>
<td>T3</td>
<td>bread, milk, peanut-butter</td>
</tr>
<tr>
<td>T4</td>
<td>beer, bread</td>
</tr>
<tr>
<td>T5</td>
<td>beer, milk</td>
</tr>
</tbody>
</table>
Interestingness of Rules

- Let’s have a look at some associations + the corresponding measures

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
<th>S</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>bread, jelly, peanut-butter</td>
<td>0.60</td>
<td>0.75</td>
</tr>
<tr>
<td>T2</td>
<td>bread, peanut-butter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>bread, milk, peanut-butter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td>beer, bread</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>beer, milk</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Support is symmetric / Confidence is asymmetric
- Confidence does not take frequency into account
Confidence vs. Conditional Probability

- Recap **Confidence (c)**
  - the strength of the association
  - \[ c = \frac{\sigma(X \cup Y)}{\sigma(X)} \]
  - (number of transactions containing all of the items in X and Y) / (number of transactions containing the items in X)
  - (support of X and Y) / (support of X)
  - conditional probability \( Pr(Y \mid X) = \frac{Pr(X \text{ and } Y)}{Pr(X)} \)

<table>
<thead>
<tr>
<th>TID</th>
<th>s</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>jelly ⇒ peanut-butter</td>
<td>0.20</td>
<td>1.00</td>
</tr>
</tbody>
</table>

“If X is bought then Y will be bought with a given probability”
→ “If jelly is bought then peanut-butter will be bought with a probability of 100%”
Apriori

- Is the most influential AR miner

- It consists of two steps
  1. Generate all frequent itemsets whose support >= minsup
  2. Use frequent itemsets to craft association rules

- Let's have a look at step one first: *Generating Itemsets*
Candidate Sets with 5 Items

null

A

B

C

D

E

AB

AC

AD

AE

BC

BD

BE

CD

CE

DE

ABC

ABD

ABE

ACD

ACE

ADE

BCD

BCE

BDE

CDE

ABCD

ABCE

ABDE

ACDE

BCDE

ABCDE
Given \( d \) unique items:
- Total number of itemsets = \( 2^d \)
- Total number of possible association rules = \( 3^d - 2^{d+1} + 1 \)

- for \( d = 5 \), there are 32 candidate item sets
- for \( d = 5 \), there are 180 rules

\[ d = 25 \rightarrow 3.4 \times 10^7 \]
\[ d = 25 \rightarrow 8.5 \times 10^{11} \]
@Generating Itemsets …

- Brute force approach is computationally expensive
  - = take all possible combinations of items
  - let’s select candidates in a smarter way

- Key idea: Downward closure property
  - any subset of a frequent itemset are also frequent itemsets

→ The algorithm iteratively does:
  - Create itemsets
  - yet, continue exploring only those whose support $\geq$ minsup
Example Itemset Generation

- discard infrequent itemsets
  - At the first level B does not meet the required support >= minsup criterion
  -> All potential itemsets that contain B can be disregarded (32 → 16)
Let's have a *Frequent Itemset* Example:

**Minimum support count = 3**

<table>
<thead>
<tr>
<th>Item</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>bread</td>
<td>4</td>
</tr>
<tr>
<td>peanut-b</td>
<td>3</td>
</tr>
<tr>
<td>jelly</td>
<td>1</td>
</tr>
<tr>
<td>milk</td>
<td>1</td>
</tr>
<tr>
<td>beer</td>
<td>1</td>
</tr>
</tbody>
</table>

**1-itemsets**

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>bread, jelly, peanut-butter</td>
</tr>
<tr>
<td>T2</td>
<td>bread, peanut-butter</td>
</tr>
<tr>
<td>T3</td>
<td>bread, milk, peanut-butter</td>
</tr>
<tr>
<td>T4</td>
<td>beer, bread</td>
</tr>
<tr>
<td>T5</td>
<td>beer, milk</td>
</tr>
</tbody>
</table>

**2-itemsets**

<table>
<thead>
<tr>
<th>Item</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>bread, peanut-b</td>
<td>3</td>
</tr>
</tbody>
</table>

Frequent Item Sets for min. support count = 3:

\{bread\}, \{peanut-b\} and \{bread, peanut-b\}
The Apriori – Algorithm(1)

- Let \( k = 1 \), set min_support
- Generate frequent itemsets of size 1
- Repeat until no new frequent itemsets are found
  - generate candidate itemsets of size \( k+1 \) from size \( k \) frequent itemsets
  - prune candidate itemsets containing subsets of size \( k \) that are infrequent
  - compute the support of each candidate by scanning the transaction DB
  - eliminate candidates that are infrequent, leaving only those that are frequent
The Apriori – Algorithm(2)

\( C_k \): Candidate itemset of size \( k \)
\( L_k \): frequent itemset of size \( k \)

\[ L_1 = \{\text{frequent items}\}; \]
\[ \text{for } (k = 1; \ L_k \neq \emptyset; \ k++) \text{ do begin} \]
\[ C_{k+1} = \text{candidates generated from } L_k; \]
\[ \text{for each transaction } t \text{ in database do} \]
\[ \text{increment the count of all candidates in } C_{k+1} \]
\[ \text{that are contained in } t \]
\[ L_{k+1} = \text{candidates in } C_{k+1} \text{ with min_support} \]
\[ \text{end} \]
\[ \text{return } \bigcup_k L_k; \]
The Apriori – Algorithm (3)

- **Join Step**
  - $C_k$ is generated by joining $L_{k-1}$ with itself

- **Prune Step**
  - Any (k-1) itemset that is not frequent cannot be a subset of a frequent k-itemset
Example of Apriori Run (1)

Minimum support count = 2

Database TDB

<table>
<thead>
<tr>
<th>Tid</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>A, C, D</td>
</tr>
<tr>
<td>20</td>
<td>B, C, E</td>
</tr>
<tr>
<td>30</td>
<td>A, B, C, F</td>
</tr>
<tr>
<td>40</td>
<td>B, E</td>
</tr>
</tbody>
</table>

1st scan

Itemset | sup
--------|-----
{A}     | 2   
{B}     | 3   
{C}     | 3   
{D}     | 1   
{E}     | 3   

L1

<table>
<thead>
<tr>
<th>Itemset</th>
<th>sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A}</td>
<td>2</td>
</tr>
<tr>
<td>{B}</td>
<td>3</td>
</tr>
<tr>
<td>{C}</td>
<td>3</td>
</tr>
<tr>
<td>{D}</td>
<td></td>
</tr>
<tr>
<td>{E}</td>
<td>3</td>
</tr>
</tbody>
</table>

2nd scan

C1

Itemset | sup
--------|-----
{A, C}  | 2   
{B, C}  | 2   
{B, E}  | 3   
{C, E}  | 2   

C2

Itemset | sup
--------|-----
{A, B}  | 1   
{A, C}  | 2   
{A, E}  | 1   
{B, C}  | 2   
{B, E}  | 3   
{C, E}  | 2   

Mark Kröll

Foundations of Knowledge Management / Association Rules
Example of Apriori Run(2)

Why not e.g. \{A, B, C\}?

→ only \{A, C\} and \{B, C\} are frequent 2-item sets
   \{A, B\} is not
→ decrease database scans
At this stage, we have all frequent itemsets

\[
\begin{array}{|c|c|}
\hline
\text{Itemset} & \text{sup} \\
\hline
\{A\} & 2 \\
\{B\} & 3 \\
\{C\} & 3 \\
\{E\} & 3 \\
\hline
\end{array}
\quad + \quad 
\begin{array}{|c|c|}
\hline
\text{Itemset} & \text{sup} \\
\hline
\{A, C\} & 2 \\
\{B, C\} & 2 \\
\{B, E\} & 3 \\
\{C, E\} & 2 \\
\hline
\end{array}
\quad + \quad 
\begin{array}{|c|c|}
\hline
\text{Itemset} & \text{sup} \\
\hline
\{B, C, E\} & 2 \\
\hline
\end{array}
\]

→ now we use these itemsets to generate association rules
Rule Generation in Apriori

- given a frequent itemset $L$
  - Find all non-empty subsets $F$ in $L$, such that the association rule $F \Rightarrow \{L-F\}$ satisfies the minimum confidence
  - Create rule $F \Rightarrow \{L-F\}$

- if $L = \{A, B, C\}$
  - The candidate itemsets are:
    - $AB \Rightarrow C$  $BC = \Rightarrow A$  $AC \Rightarrow B$
    - $C \Rightarrow AB$  $A = \Rightarrow BC$  $B \Rightarrow AC$

  - In general, there are $2^k - 2$ candidate solutions,
    - where $k$ is the size of itemset $L$
Can we be more efficient?

- can we apply the same heuristic used with itemset generation?
  - Confidence does not have anti-monotone property
  - That is, \( c(AB=>D) > c(A=>D) \)?
    - We don’t know …

- but confidence of rules generated from the same itemset does have the anti monotone property
  - \( L = \{A, B, C, D\} \)
    - \( c(ABC =>D) >= c(AB=>CD) >= c(A=>BCD) \)
  - We can use this property to inform the rule generation
Example of Efficient Rule Generation

Frequent Itemset \{A, B, C, D\}

Diagram showing association rules and frequent itemsets.
@Quality of Generated Rules

- Apriori algorithm produces a lot of rules
  - many of them redundant
  - many of them uninteresting
  - many of them uninterpretable

- Strong Rules can be misleading
  - strong = high support and/or high confidence
  - yet, not all strong association rules are interesting enough to be presented and used (see next slide for an example)

→ If a rule is not interpretable or intuitive in the face of domain-specific knowledge, it need not be adopted and used for decision-making purposes.
Strong Rules Are Not Necessarily Interesting (1)

- Example from [Aggarwal & Yu, PODS98]
  - among 5000 students
  - 3000 play basketball (=60%), 3750 eat cereal (=75%), 2000 both play basket ball and eat cereal (=40%)
  - minsup (40%) and minconf (60%)

- Rule play basketball ⇒ eat cereal [s= 40%, c = 66.7%]
  is misleading because the overall percentage of students eating cereal is 75% which is higher than 66.7%

\[ \text{P(eat cereal)} > \text{P(eat cereal | play basketball)} \]
\[ \begin{align*}
0.75 & \quad 0.66 \\
\end{align*} \]

⇒ negative association ("playing basketball" decreases "eat cereal")
Strong Rules Are Not Necessarily Interesting (2)

- statistical (linear) independence test (e.g. correlation)
  - Heuristics to measure association
    
    \( A \Rightarrow B \) is interesting if

    \[
    \frac{\text{support}(A, B)}{\text{support}(A)} - \text{support}(B) > d
    \]
    
    or,

    \[
    \text{support}(A, B) - [\text{support}(A) \cdot \text{support}(B)] > k
    \]

- example: the association rule in the previous example
  - support(play basketball, eat cereal) -
    
    \[
    \frac{\text{support}(\text{play basketball}) \cdot \text{support}(\text{eat cereal})}{\text{support}(\text{play basketball}) \cdot \text{support}(\text{eat cereal})}
    \]
  
  \[
  = 0.4 - [0.6 \cdot 0.75]
  \]
  
  \[
  = -0.05 < 0 \text{ (negative associated !)}
  \]
Limitations of Apriori

- **Bottlenecks:**
  - Apriori scans the transaction DB several times
  - usually, there is a large number of candidates
  - calculation of candidate’s *support count* can be time-consuming

- **Improvements:**
  - reduce the number of DB scans
  - shrink the number of candidates
  - more efficient support counting for candidates
Revisiting Candidate Generation

- **Remember**
  - Use the previous frequent itemsets (k-1) to generate the k-itemsets
  - Count itemsets support by scanning the database

- **Bottleneck in the process: Candidate Generation**
  - Suppose 100 items
  - First level of the tree → 100 nodes
    - Second level of the tree → \( \binom{100}{2} \)
  - In general, number of k-itemsets: \( \binom{100}{k} \)
Avoid Candidate Generation

- build auxiliary structure (Frequent Pattern Tree)
  - to get statistics about the itemsets to avoid candidate generation
  - avoid multiple scans of the data

<table>
<thead>
<tr>
<th>TID</th>
<th>Items bought</th>
<th>(ordered) frequent items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>{f,a,c,d,g,i,m,p}</td>
<td>{f,c,a,m,p}</td>
</tr>
<tr>
<td>200</td>
<td>{a,b,c,f,l,m,o}</td>
<td>{f,c,a,b,m}</td>
</tr>
<tr>
<td>300</td>
<td>{b,f,h,j,o}</td>
<td>{f,b}</td>
</tr>
<tr>
<td>400</td>
<td>{b,c,k,s,p}</td>
<td>{c,b,p}</td>
</tr>
<tr>
<td>500</td>
<td>{a,f,c,e,l,p,m,n}</td>
<td>{f,c,a,m,p}</td>
</tr>
</tbody>
</table>

→ quick access to nodes of the tree

for further information see:
- (Jiawei Han, Jian Pei, Yiwen Yin: Mining Frequent Patterns without Candidate Generation In Proceedings of the 2000 ACM SIGMOD international Conference on Management of Data.)
FP-growth vs. Apriori: Einfluss von $s_{\text{min}}$

- FP-growth is about a magnitude faster than Apriori because of
  - no candidate generation and testing
  - more compact data structure
  - no iterative database scans
Hierarchical Association Rules

Problem with plain itemsets (parameter setting):
- High minsup: apriori finds only few rules
- Low minsup: apriori finds unmanagably many rules

→ exploit item taxonomies (generalizations, is-a hierarchies) which exist in many applications

Objective: find association rules between generalized items
- support for sets of item types (e.g., product groups) is higher than support for sets of individual items
Motivation

- **Examples:**
  - jeans => boots
  - jackets => boots
  - outerwear => boots
  
  - support < minsup
  - support > minsup

- **Characteristics:**
  - support ("outerwear => boots") is not necessarily equal to the sum support("jeans => boots") + support ("jackets => boots")
  
  - If the support of rule "outerwear => boots" exceeds minsup, then the support of rule "clothes => boots" does too
Example

<table>
<thead>
<tr>
<th>transaction id</th>
<th>items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>shirt</td>
</tr>
<tr>
<td>2</td>
<td>jacket, boots</td>
</tr>
<tr>
<td>3</td>
<td>jeans, boots</td>
</tr>
<tr>
<td>4</td>
<td>sports shoes</td>
</tr>
<tr>
<td>5</td>
<td>sports shoes</td>
</tr>
<tr>
<td>6</td>
<td>jacket</td>
</tr>
</tbody>
</table>

- Support of \{clothes\}: 4 of 6 = 67%
- Support of \{clothes, boots\}: 2 of 6 = 33%
- “shoes => clothes”: support 33%, confidence 50%
- “boots => clothes”: support 33%, confidence 100%

- Procedure:
  - replace items by items located higher in the hierarchy
  - apply Apriori
Types of Association Rules

- **Binary Association Rules**
  - Bread => Peanut Butter

- **Quantitative Association Rules**
  - numeric attributes
  - Weight in [70kg – 90kg] => height in [170cm – 190cm]

- **Fuzzy Association Rules**
  - allow different degrees of membership (several categories)
  - to overcome the sharp boundary problem

- In this lecture, we focused on **Binary Association Rules**
Other Application Areas of AR

- Analysis of credit card purchases.
  - identify the most influential factors common to non-profitable customers, e.g. credit card limit, etc.

- Identification of fraudulent medical insurance claims.
  - analyse claim forms submitted by patients to a medical insurance company
  - find relationships among medical procedures that are often performed together
  - might be indicative for fraudulent behavior, when common rules are broken

- Recommendation Systems
  - E.g. Amazon’s “Customers who bought this item also bought”
  - … is based on association rules
Available Toolkits

- **WEKA**
  - freely available library implemented in Java
  - provides variants of the Apriori Algorithm

- **R**
  - [http://www.r-project.org/](http://www.r-project.org/)

- **DBMiner System**
  - [Han et al. 1996]
Summary of Today’s Lecture (1)

- **Association Rules** represent an unsupervised learning method
  - that attempts to capture associations between groups of items

- **Association Rules** are “if-then rules” with two measures
  - which quantify the support and confidence of the rule

  \[ X \implies Y \]

  - = if items in group X appear in a market basket what is the probability that items in group Y will also be purchased?

- **Association rule mining** is also known as
  - frequent item set mining
  - market basket analysis
  - affinity analysis
Summary of Today’s Lecture (2)

- **Apriori is most influential rule miner**
  - Consisting of two steps:
    1. Generating Frequent Itemsets
    2. Generating Association Rules from these sets

- **Challenges/ Improvements**
  - exponential runtime / efficient data structures (FP – tree)
  - rule quality / metrics: interestingness of rules

- **Further Directions:**
  - Application to sequences in order to look for patterns that evolve over time
Thanks for your attention!

Any Questions …