Grouping Association Rules Using Lift

Michael Hahsler

Intelligent Data Analysis Group, Southern Methodist University

11th INFORMS Workshop on Data Mining and Decision Analytics
November 12, 2016

SMU | Bobby B. Lyle School of Engineering
Table of Contents

1 Background

2 Motivation

3 Clustering Association Rules

4 Grouping Rules Using Lift

5 Illustrative Application

6 Conclusion
Association Rules

Mining association rules was first introduced by Agrawal et al. (1993) as:

- Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of $n$ binary attributes called items.
- Let $D = \{t_1, t_2, \ldots, t_m\}$ be a set of transactions called the database. Each transaction in $D$ contains a subset of the items in $I$.
- An itemset $X$ is a subset of $I$.
- A rule is defined as an implication of the form

$$X \Rightarrow Y$$

where $X$ and $Y$ are itemsets.
**Association Rules II**

- **Support:** \( \text{supp}(X) \) is proportion of transactions which contain \( X \)
- **Confidence:** \( \text{conf}(X \Rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)} \)

**Association rule** \( X \Rightarrow Y \) needs to satisfy:

\[
\text{supp}(X \cup Y) \geq \sigma \quad \text{and} \quad \text{conf}(X \Rightarrow Y) \geq \delta
\]

**Example**

\{milk, bread\} \Rightarrow \{butter\}

- support = 0.2
- confidence = 0.9
- lift = 2
The AR Mining Process

Two-step process

1. Minimum support is used to generate the set of all frequent itemsets.
2. Each frequent itemsets is used to generate all possible rules which satisfy the minimum confidence constraint.
The AR Mining Process

Two-step process

1. Minimum support is used to generate the set of all frequent itemsets.
2. Each frequent itemsets is used to generate all possible rules which satisfy the minimum confidence constraint.

Worst case: $2^n - n - 1$ frequent itemsets (size $\geq 2$ for $n$ distinct items). Each frequent generates $2^+ \text{ rules} \Rightarrow O(2^n)$. 
The AR Mining Process

Two-step process

1. Minimum support is used to generate the set of all frequent itemsets.
2. Each frequent itemsets is used to generate all possible rules which satisfy the minimum confidence constraint.

Worst case: $2^n - n - 1$ frequent itemsets (size $\geq 2$ for $n$ distinct items). Each frequent generates $2^+\text{ rules} \Rightarrow O(2^n)$.

Practical Strategy

Increase minimum support to reduce number of rules $\Rightarrow$ misses important rules.

We need to be able to deal with large sets of association rules.
Motivation

- Association rule mining is a popular data mining method.
- It is known to produce large sets of rules.
- Clustering is a well known data reduction method.
Motivation

- Association rule mining is a popular data mining method.
- It is known to produce large sets of rules.
- Clustering is a well known data reduction method.

Questions

- Why is association rule clustering not a standard functionality in data mining tools?
- Why were only so few papers published clustering association rules? Lent et al. (1997); Gupta et al. (1999); Toivonen et al. (1995); Adomavicius and Tuzhilin (2001); An et al. (2003); Berrado and Runger (2007)
<table>
<thead>
<tr>
<th>Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>
The Clustering Problem

**Goal**

Group a set of $m$ association rules

$$\mathcal{R} = \{R_1, R_2, \ldots, R_m\}$$

into $k$ subsets

$$\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$$

called clusters.

Rules in the same cluster should be more similar to each other than to rules in different clusters.
Clustering Binary Vectors

A set of $m$ association rules $\mathcal{R}$ can be represented as a set of $n$-dimensional vectors

$$x_1, x_2, \ldots, x_m$$

where $n$ is the total number of different items in the database. $i = 1, 2, \ldots, m$.

Example: Rules and Binary Representation

<table>
<thead>
<tr>
<th>lhs</th>
<th>rhs</th>
<th>support</th>
<th>confidence</th>
<th>lift</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1] {tropical fruit, root vegetables} =&gt; {other vegetables}</td>
<td>0.0123</td>
<td>0.585</td>
<td>3.02</td>
<td></td>
</tr>
<tr>
<td>[2] {tropical fruit, root vegetables} =&gt; {whole milk}</td>
<td>0.0120</td>
<td>0.570</td>
<td>2.23</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>tropical fruit root vegetables other vegetables whole milk</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,] 1 1 1 0</td>
</tr>
<tr>
<td>[2,] 1 1 0 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>yogurt rolls/buns bottled water soda</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1,] 0 0 0 0</td>
</tr>
<tr>
<td>[2,] 0 0 0 0</td>
</tr>
</tbody>
</table>
**k-Means Problem**

Find a cluster assignment $S = \{S_1, S_2, \ldots, S_k\}$ which minimizes

$$WSS = \sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - \mu_i||^2,$$

where $\mu_i$ is the cluster centroid.

**Advantage:**
- Fast and efficient heuristics.

**Disadvantage:**
- Implies Euclidean distance, but matching 1s (same items in the rule) are much more important than matching 0s.
Jaccard Index

Let $X_i$ and $X_j$ be the set of all items contained in $R_i$ and $R_j$,

$$d_{Jaccard}(X_i, X_j) = 1 - \frac{|X_i \cap X_j|}{|X_i \cup X_j|}.$$ 

I.e., number of items they have in common divided by the number of unique items in both sets.

Can be used in hierarchical and other clustering techniques.
Jaccard Index

Let $X_i$ and $X_j$ be the set of all items contained in $R_i$ and $R_j$,

$$d_{\text{Jaccard}}(X_i, X_j) = 1 - \frac{|X_i \cap X_j|}{|X_i \cup X_j|}.$$  

I.e., number of items they have in common divided by the number of unique items in both sets.
Can be used in hierarchical and other clustering techniques.

**Comparison**

<table>
<thead>
<tr>
<th>Rules</th>
<th>$d_E$</th>
<th>$d_J$</th>
</tr>
</thead>
<tbody>
<tr>
<td>${\text{bread}} \rightarrow {\text{butter}}$</td>
<td>2</td>
<td>1.42</td>
</tr>
<tr>
<td>${\text{beer}} \rightarrow {\text{liquor}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>${\text{bread}, \text{milk}, \text{cheese}} \rightarrow {\text{butter}}$</td>
<td>2</td>
<td>0.67</td>
</tr>
<tr>
<td>${\text{bread}, \text{vegetables}, \text{yogurt}} \rightarrow {\text{butter}}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Issue:** Very sparse binary data.
Common Covered Transactions

Toivonen et al. (1995) define the distance between two rules with a common consequent, $X \rightarrow Z$ and $Y \rightarrow Z$, as

$$d_{Toivonen}(X \rightarrow Z, Y \rightarrow Z) = |m(X \cup Z)| + |m(Y \cup Z)| - 2|m(X \cup Y \cup Z)|,$$

where $m(X)$ is the set of transactions in $D$ which are covered by the rule, i.e.,

$$m(X) = \{ t \mid t \in D \land X \subseteq t \}.$$

Computes the number of transactions which are covered only by one of the rules but not by both.
Gupta et al. (1999) define for \( X_i \) and \( X_j \), the sets of all items in two rules, the distance as

\[
d_{\text{Gupta}}(X_i, X_j) = 1 - \frac{|m(X_i \cup X_j)|}{|m(X_i)| + |m(X_j)| - |m(X_i \cup X_j)|}
\]

Proportion of transactions which are covered by both rules in the transactions which are covered by at least one of the rules.

**Advantage:**
- Avoids the problems of clustering sparse, high-dimensional binary vectors.

**Disadvantage:**
- Introduces a strong bias towards clustering subsets.
Adomavicius and Tuzhilin (2001) use the item hierarchy:

1. Select an appropriate level in the hierarchy.
2. Replacing each item in all rules by the label of the group it belongs to.
3. Rules which are now exactly the same will be grouped.

Example: The two rules 
\{butter\} → \{bread\} and 
\{cream, cheese\} → \{beagles\} are both grouped at the product group level as 
\{dairy\} → \{baked goods\}.

This reduces problems with high dimensionality and sparseness. Groups substitutes (e.g., bread and beagles) if they are in the same subtree. Rules have to match exactly to be grouped.
Adomavicius and Tuzhilin (2001) use the item hierarchy:

1. Select an appropriate level in the hierarchy.
2. Replacing each item in all rules by the label of the group it belongs to.
3. Rules which are now exactly the same will be grouped.

**Example**

The two rules \{butter\} → \{bread\} and \{cream\} \{cheese\} → \{beagles\} are both grouped at the product group level as \{dairy\} → \{baked\} \{goods\}.
Adomavicius and Tuzhilin (2001) use the item hierarchy:

1. Select an appropriate level in the hierarchy.
2. Replacing each item in all rules by the label of the group it belongs to.
3. Rules which are now exactly the same will be grouped.

Example

The two rules \{butter\} \rightarrow \{bread\} and \{cream cheese\} \rightarrow \{beagles\} are both grouped at the product group level as \{dairy\} \rightarrow \{baked goods\}.

- Reduces problems with high dimensionality and sparseness.
- Groups substitutes (e.g., bread and beagles) if they are in the same subtree.
- Rules have to match exactly to be grouped.
High dimensionality and sparseness: Binary vectors are extremely high-dimensional and sparse.

Substitutes: Grouping rules with substitutes, e.g., bread and beagles, is important.

Direction of association: Most approaches do not differentiate between LHS and RHS.

Computational Complexity: Distance matrix for a set of $m$ rules requires $O(m^2)$ time and space.

Frequent itemset structure: Clustering association rules will just rediscover subset structure of the frequent itemset lattice.
Frequent Itemset Structure

'{beer, eggs, flour, milk}  support count = 0

{beer, eggs, flour} 1  {beer, eggs, milk} 0  {beer, flour, milk} 0  {eggs, flour, milk} 2

{beer, eggs} 1  {beer, flour} 1  {beer, milk} 0  {eggs, flour} 3  {eggs, milk} 2  {flour, milk} 2

{beer} 1  {eggs} 4  {flour} 3  {milk} 4

'Frequent Itemsets'
Frequent Itemset Structure

Clusters:
- **Cluster 1**: {beer} 1, {beer, eggs} 1, {beer, flour} 1, {beer, milk} 0, {eggs} 4, {flour} 3
- **Cluster 2**: {milk} 4, {eggs, milk} 2, {flour, milk} 2
- **Cluster 3**: {eggs, flour, milk} 2

Frequent Itemsets:

- {flour} 3, {beer} 1, {eggs} 4, {milk} 4
- {beer, eggs} 1, {beer, flour} 1, {beer, milk} 0, {eggs, flour} 3, {eggs, milk} 2, {flour, milk} 2
- {beer, eggs, flour, milk} support count = 0

M. Hahsler (IDA@SMU)
Brin et al. (1997) introduced lift as

\[
\text{lift}(X \Rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)\text{supp}(Y)}
\]

- Deviation of independence of LHS and RHS.
- 1 indicates independence.
- Larger lift values (\(\gg 1\)) indicate association.
Grouping Rules Using Lift

Brin et al. (1997) introduced lift as

\[
\text{lift}(X \Rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)\text{supp}(Y)}
\]

- Deviation of independence of LHS and RHS.
- 1 indicates independence.
- Larger lift values (\(\gg 1\)) indicate association.

**Idea**

Rules with a LHS that have strong dependencies with the same set of RHS (i.e., have a high lift value) are similar and thus should be grouped together.
Grouping Rules Using Lift

Brin et al. (1997) introduced lift as

\[
\text{lift}(X \Rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)\text{supp}(Y)}
\]

- Deviation of independence of LHS and RHS.
- 1 indicates independence.
- Larger lift values ($\gg 1$) indicate association.

Idea

Rules with a LHS that have strong dependencies with the same set of RHS (i.e., have a high lift value) are similar and thus should be grouped together.

Example

If \{butter, cheese\} $\rightarrow$ \{bread\} and \{margarine, cheese\} $\rightarrow$ \{bread\} have a similarly high lift, then the LHS should be grouped.

Note: butter and margarine are substitutes!
Definition

\[ \mathcal{R} = \{ \langle X_1, Y_1, \theta_1 \rangle, \ldots, \langle X_i, Y_i, \theta_i \rangle, \ldots, \langle X_n, Y_n, \theta_n \rangle \} \]

- where \( X_i \) is the LHS,
- \( Y_i \) is the RHS and
- \( \theta_i \) is the lift value for the \( i \)-th rule.
Definition

\[ \mathcal{R} = \{ \langle X_1, Y_1, \theta_1 \rangle, \ldots, \langle X_i, Y_i, \theta_i \rangle, \ldots, \langle X_n, Y_n, \theta_n \rangle \} \]

- where \( X_i \) is the LHS,
- \( Y_i \) is the RHS and
- \( \theta_i \) is the lift value for the \( i \)-th rule.

Process

1. Find \( A \), the set of unique LHS and \( C \), the unique RHS.
2. Create a \( A \times C \) matrix \( \mathbf{M} = (m_{ac}) \).
3. Populate with \( m_{ac} = \theta_i \) where \( X_i \) has index \( a \) in \( A \) and \( Y_i \) has index \( c \) in \( C \).
4. Impute missing values (we use a neutral lift value of 1).
5. Cluster rules by grouping columns and/or rows in \( \mathbf{M} \).
Grouping LHS

We define now the distance between two LHS $X_i$ and $X_j$ as the Euclidean distance

$$d_{\text{Lift}}(X_i, X_j) = ||m_i - m_j||,$$

where $m_i$ and $m_j$ are the column vectors representing all rules with the LHS of $X_i$ and $X_j$, respectively.

We can use now hierarchical clustering, $k$-medoids or $k$-means. For efficiency reasons we use a $k$-means heuristic to minimize the WSS

$$\arg\min_S \sum_{i=1}^{k} \sum_{m_j \in S_i} ||m_j - \mu_i||^2,$$

Most tools create rules with a single item in the RHS $\rightarrow$ no need for grouping.
Table of Contents

1. Background
2. Motivation
3. Clustering Association Rules
4. Grouping Rules Using Lift
5. Illustrative Application
6. Conclusion
Example: Create Rules

```r
R> library("arules")
R> library("arulesViz")
R> data("Groceries")
R> Groceries

transactions in sparse format with
9835 transactions (rows) and
169 items (columns)

R> rules <- apriori(Groceries, parameter=list(support=0.001, confidence=0.5), + control=list(verbose=FALSE))
R> rules

set of 5668 rules

R> inspect(head(sort(rules, by="lift"),3))

  lhs                                      rhs                          support
[1] {Instant food products,soda} => {hamburger meat} 0.00122
[2] {soda,popcorn} => {salty snack} 0.00122
[3] {flour,baking powder} => {sugar} 0.00102

  confidence lift
[1] 0.632 19.0
[2] 0.632 16.7
[3] 0.556 16.4
```
Example: Create Rules

R> plot(rules, method="grouped", control = list(gp_labels= gpar(cex=1), main = ""))
Table of Contents

1 Background

2 Motivation

3 Clustering Association Rules

4 Grouping Rules Using Lift

5 Illustrative Application

6 Conclusion
Conclusion

Main Advantages

- Avoids high dimensionality and sparsity.
- Handles (relatively) large rule sets.
- Can group substitute items.
- Visualization guides the user automatically to the most interesting groups/rules.
- Easy to understand (similar to matrix-based visualization)

Code

Association rule mining and clustering is implemented in the R extension package `arules` (Hahsler et al., 2005). Grouping by lift and visualizations are available in the extension package `arulesViz` (Hahsler and Chelluboina, 2016). Both are freely available from the Comprehensive R Archive Network at

http://CRAN.R-project.org/package=arules.


