Association rules

Data Mining

Laboratory

Ewa Figielska

Warsaw School of Computer Science

Affinity analysis

- **Affinity analysis** is the study of attributes or characteristics that "go together."
- Methods for affinity analysis (also known as **market basket analysis**) seek to uncover **associations** among attributes.
- **Association rules** take the form:

  "If antecedent, then consequent,"

  along with a measure of the **support** and **confidence** associated with the rule.

- Example.

  A particular supermarket may find that of the 1000 customers shopping on a Thursday night, 200 bought diapers, and 50 of them bought beer.

  The association rule is:

  "If buy diapers, then buy beer,"

  with a support of 50/1000 = 5% and a confidence of 50/200 = 25%.

Examples of association tasks

- Examining the proportion of children whose parents read to them who are themselves good readers
- Predicting degradation in telecommunications networks
- Finding out which items in a supermarket are purchased together, and which items are never purchased together
- Determining the proportion of cases in which a new drug will exhibit dangerous side effects

Problems with creating association rules

- The number of possible association rules grows exponentially with the number of attributes.
- If there are *k* attributes (limited o to binary attributes, buy beer = yes, buy beer = no) there are on the order of $k2^{k-1}$possible association rules.

- Example.

  For three items a, b, c, there are $3 \cdot 2^2 = 12$ rules

Rule #		Rule #	
1	a -> b	7	a, b -> c
2	a -> c	8	a, c -> b
3	b -> a	9	b, c -> a
4	b -> c	10	a -> b, c
5	c -> a	11	b -> a,c
6	c -> b	12	c -> a, b

- Typically there may be thousands of binary attributes (*buy beer? buy popcorn? buy milk? buy bread?* etc.)
- Example.

  Suppose that a tiny store has 100 different items, and a customer could either buy or not buy any combination of those 100 items. So, there are $100 \cdot 2^{99} \cong 6.4 \cdot 10^{31}$ possible association rules.

- The **a priori algorithm** for mining association rules takes advantage of structure within the rules themselves to reduce the search problem to a more manageable size.

Example.Data representation.

- Set I of 7 items: bread, butter, cheese, honey, milk, sugar and tea.

transaction #	bread	butter	cheese	honey	milk	sugar	tea
1	0	0	1	1	1	0	0
2	1	0	0	1	0	1	0
3	0	1	0	1	0	1	1
4	0	1	0	1	1	0	1
5	1	1	1	0	0	0	0
6	1	1	0	0	0	1	1
7	0	0	0	1	0	0	1
8	0	0	1	0	1	0	1
9	1	1	0	0	0	1	0
10	0	1	0	1	0	0	0
11	0	1	1	0	1	1	0
12	1	1	0	0	0	1	0
13	1	1	0	1	0	1	0
14	0	1	1	1	1	0	1

Tabular data format

Support, confidence

- Let D be the set of transactions.
- Each transaction T in D represents a set of items contained in I (I is the set of items).
- Suppose that we have a particular set of items A (e.g. butter and sugar), and another set of items B (e.g. bread). An **association rule** takes the form:

  ***if A, then B* (i.e. *A* ⇒ *B*),**

  where the **antecedent** A and the **consequent** B are proper subsets of I , and A and B are mutually exclusive.

- The **support**, ***s***, for a particular association rule $A \Rightarrow B$:

$$s = P(A \cap B) = \frac{\text{number of transactions containing both } A \text{ and } B}{\text{total number of transactions}}$$

- The **confidence**, ***c***, of the association rule $A \Rightarrow B$ is a measure of the accuracy of the rule:

$$c = P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{\text{number of transactions containing both } A \text{ and } B}{\text{number of transactions containing } A}$$

- **Strong rules** = rules which meet or surpass certain minimum support and confidence criteria.

Frequent itemsets.

- An **itemset** is a set of items contained in I .
- The ***k*-itemset** is an itemset containing *k* items.
- The **itemset frequency** is the number of transactions that contain the particular itemset.
- A **frequent itemset** is an itemset that occurs at least a certain minimum number of times, having itemset frequency $\geq \Phi$.

- The itemsets that occur Φ and more than Φ times are said to be **frequent**.
- The set of frequent *k*-itemsets is denoted as $\boldsymbol{F_k}$.

Mining association rules.

- The mining of association rules from large databases is a two-steps process:

  1. Find all frequent itemsets (i.e. find all itemsets with frequency ≥ Φ).

  2. From the frequent itemsets, generate association rules satisfying the minimum support and confidence conditions.

- **A PRIORI PROPERTY**

  If an itemset Z is not frequent then for any item A, $Z \cup A$ will not be frequent.

- The **a priori algorithm** takes advantage of the a priori property to shrink the search space.

Example. Algorithm a priori. Generating frequent itemsets

- Let $\Phi = 4$.
- F_1 is the set of the frequent 1-itemsets; it represents the individual items themselves.

  $F_1 = \{\{bread\}, \{butter\}, \{cheese\}, \{honey\}, \{milk\}, \{sugar\}, \{tea\}\}$.

itemset	count
bread	6
butter	10
cheese	5
honey	8
milk	5
sugar	7
tea	6

- F_2 is the set of the frequent 2-itemsets.
- $F_2 = \{\{bread, butter\},\ \{bread, sugar\},\ \{butter, honey\},\ \{butter, sugar\},\ \{butter, tea\},\ \{cheese, milk\},\ \{honey, tea\}\}$.

itemset	cout	itemset	count
bread, butter	5	cheese, honey	2
bread, cheese	1	cheese, milk	4
bread, honey	2	cheese, sugar	1
bread, milk	0	cheese, tea	2
bread, sugar	5	honey, milk	3
bread, tea	1	honey, sugar	3
butter, cheese	3	honey, tea	4
butter, honey	5	milk, sugar	1
butter, milk	3	milk, tea	3
butter, sugar	6	sugar, tea	2
butter, tea	4		

Example. Generating frequent itemsets

itemset	cout	itemset	count
bread, butter	5	cheese, honey	2
bread, cheese	1	cheese, milk	4
bread, honey	2	cheese, sugar	1
bread, milk	0	cheese, tea	2
bread, sugar	5	honey, milk	3
bread, tea	1	honey, sugar	3
butter, cheese	3	honey, tea	4
butter, honey	5	milk, sugar	1
butter, milk	3	milk, tea	3
butter, sugar	6	sugar, tea	2
butter, tea	4		

In general, to find F_k , the a priori algorithm:

- constructs a set C_k of candidate k-itemsets by joining F_{k-1} with itself,
  - e.g. C_3 of candidate 3- itemsets is constructed by joining itemstests from F_2 if they have the first $2-1=\ 1$ items in common;
  - **in general, itemsets from $\boldsymbol{F_n}$ are joined if they have the first $\boldsymbol{n-1}$ items in common** (in alphabetical order).
- prunes C_k using the a priori property;

$F_2 = \{\{bread, butter\},\ \{bread, sugar\},\ \{butter, honey\},\ \{butter, sugar\},\quad \{butter, tea\},$
$\{cheese, milk\},\ \{honey, tea\}\}.$
$C_3\ =\ \{\{bread, butter, sugar\},\ \ \{butter, honey, sugar\},\ \ \{butter, honey, tea\},$
$\{butter, sugar, tea\}\}$

Pruning C_3 using the a priori property.

- For each itemset t in C_3, its subsets of size 2 (i.e. $k\ -\ 1$) are generated and examined.
- If any of these subsets are not frequent, t cannot be frequent and is therefore pruned.

Pruned itemsets:

- $\{butter, honey, sugar\}$ because $\{honey, sugar\}$ is not frequent
- $\{butter, sugar, tea\}$ because $\{sugar, tea\}$ is not frequent

The count of the remaining sets is checked:

- $\{bread, butter, sugar\}$ count $=\ 4=\Phi$, thus $F_3\ =\ \{bread, butter, sugar\}$
- $\{butter, honey, tea\}$ count $=\ 3\leq\Phi$; this itemset is pruned

Generating association rules

1. For each frequent itemset t, generate all subsets of t.
2. Let tt represent a nonempty subset of t. Consider the association rule R : $tt \Rightarrow (t - tt)$, where $(t - tt)$ indicates the set t without tt. Generate R if R fulfills the minimum confidence requirement. Do so for every subset tt of t (note that for simplicity, a single-item consequent is often desired).

- Consider F_3

  $F_3 = \{bread, butter, sugar\}$

  The proper subsets of $t = \{bread, butter, sugar\}$:

  $\{bread\}, \{butter\}, \{sugar\}, \{bread, butter\}, \{bread, sugar\}, \{butter.sugar\}$

  Candidate association rules with two antecedents

  1. **{bread,butter} ⇒ {sugar}; s = 4/14 = 28.6%; c = 4/5 = 80%**
  2. **{bread,sugar} ⇒ {butter}; s = 4/14 = 28.6%; c = 4/5 = 80%**
  3. {butter,sugar} ⇒ {bread}; s = 4/14 = 28.6%; c = 4/6 = 66.7%

itemset	cout	itemset	count
bread, butter	5	cheese, honey	2
bread, cheese	1	cheese, milk	4
bread, honey	2	cheese, sugar	1
bread, milk	0	cheese, tea	2
bread, sugar	5	honey, milk	3
bread, tea	1	honey, sugar	3
butter, cheese	3	honey, tea	4
butter, honey	5	milk, sugar	1
butter, milk	3	milk, tea	3
butter, sugar	6	sugar, tea	2
butter, tea	4		

itemset	count
bread	6
butter	10
cheese	5
honey	8
milk	5
sugar	7
tea	6

$$s = P(A \cap B) \qquad c = P(B|A) = \frac{P(A \cap B)}{P(A)}$$

Generating association rules

- Consider F_2

  $F_2 = \{\{bread, butter\}, \{bread, sugar\}, \{butter, honey\}, \{butter, sugar\}, \{butter, tea\}, \{cheese, milk\}, \{honey, tea\}\}.$

  The proper subsets of $t = \{bread, butter\}$: $\{bread\}, \{butter\},$ and so on...

  Candidate association rules:

  1. **{bread} ⇒ {butter}; s = 5/14 = 35.7%; c = 5/6 = 83.3%**
  2. {butter} ⇒ {bread}; s = 5/14 = 35.7 %; c = 5/10 = 50%
  3. **{bread} ⇒ {sugar}; s = 5/14 = 35.7%; c = 5/6 = 83.3%**
  4. {sugar} ⇒ {bread}; s = 5/14 = 35.7 %; c = 5/7 = 71.4%
  5. {butter} ⇒ {honey}; s = 5/14 = 35.7 %; c = 5/10 = 50%
  6. {honey} ⇒ {butter}; s = 5/14 = 35.7 %; c = 5/8 = 62.5%
  7. {butter} ⇒ {sugar}; s = 6/14 = 42.9%; c = 6/10 = 60%
  8. **{sugar} ⇒ {butter}; s = 6/14 = 42.9%; c = 6/7 = 85.7%**
  9. {butter} ⇒ {tea}; s = 4/14 = 28.6%; c = 4/10 = 40%
  10. {tea} ⇒ {butter}; s = 4/14 = 28.6%; c = 4/6 = 66.7%
  11. **{cheese} ⇒ {milk}; s = 4/14 = 28.6%; c = 4/5 = 80%**
  12. **{milk} ⇒ {cheese}; s = 4/14 = 28.6%; c = 4/5 = 80%**
  13. {honey} ⇒ {tea}; s = 4/14 = 28.6%; c = 4/ 8= 50%
  14. {tea} ⇒ {honey}; s = 4/14 = 28.6%; c = 4/6 = 66.7%

itemset	count
bread	6
butter	10
cheese	5
honey	8
milk	5
sugar	7
tea	6

itemset	cout	itemset	count
bread, butter	5	cheese, honey	2
bread, cheese	1	cheese, milk	4
bread, honey	2	cheese, sugar	1
bread, milk	0	cheese, tea	2
bread, sugar	5	honey, milk	3
bread, tea	1	honey, sugar	3
butter, cheese	3	honey, tea	4
butter, honey	5	milk, sugar	1
butter, milk	3	milk, tea	3
butter, sugar	6	sugar, tea	2
butter, tea	4		

Example in R language

1. `install.packages("arules")`
2. `library(arules)`
3. `td <- read.transactions('mstore1.csv', sep=',')` # read file as transactions,usual read.csv() won't do, as it expects equal number of data points per row

Data file with transactions

```
cheese,honey,milk
bread,honey,sugar
butter,honey,sugar,tea
butter,honey,milk,tea
bread,butter,cheese
bread,butter,sugar,tea
honey,tea
cheese,milk,tea
bread,butter,sugar
butter,honey
butter,cheese,milk,sugar
bread,butter,sugar
bread,butter,honey,sugar
butter,cheese,honey,milk,tea
```

4. `arules <- apriori(td,parameter=list(supp=0.2,conf=.80, minlen=2, maxlen=3, target='rules'))` # run a priori algorithms
5. `inspect(arules)` #show the rules
6. `inspect(sort(arules, by="confidence", decreasing=TRUE))` #sort the rules

Output

```
> inspect(sort(rules, by="confidence", decreasing=TRUE))
    lhs               rhs      support   confidence lift
[1] {sugar}        => {butter} 0.4285714 0.8571429  1.200000
[2] {bread}        => {sugar}  0.3571429 0.8333333  1.666667
[3] {bread}        => {butter} 0.3571429 0.8333333  1.166667
[4] {cheese}       => {milk}   0.2857143 0.8000000  2.240000
[5] {milk}         => {cheese} 0.2857143 0.8000000  2.240000
[6] {bread,sugar}  => {butter} 0.2857143 0.8000000  1.120000
[7] {bread,butter} => {sugar}  0.2857143 0.8000000  1.600000
```

Example using tabular data format

1. `d<-read.csv("mstore2.csv")`
2. `rules<-apriori(d, parameter = list(supp = 0.2, conf=0.80, maxlen=3, target = 'rules'))`
3. `inspect(rules)`

```
> inspect(rules)
     lhs            rhs          support   confidence lift
[1]  {cheese=y}  => {milk=y}     0.2857143 0.8000000  2.240000
[2]  {milk=y}    => {cheese=y}   0.2857143 0.8000000  2.240000
[3]  {cheese=y}  => {sugar=n}    0.2857143 0.8000000  1.600000
[4]  {cheese=y}  => {bread=n}    0.2857143 0.8000000  1.400000
[5]  {milk=y}    => {sugar=n}    0.2857143 0.8000000  1.600000
[6]  {milk=y}    => {bread=n}    0.3571429 1.0000000  1.750000
[7]  {bread=y}   => {sugar=y}    0.3571429 0.8333333  1.666667
[8]  {bread=y}   => {tea=n}      0.3571429 0.8333333  1.458333
[9]  {bread=y}   => {cheese=n}   0.3571429 0.8333333  1.296296
[10] {bread=y}   => {milk=n}     0.4285714 1.0000000  1.555556
[11] {bread=y}   => {butter=y}   0.3571429 0.8333333  1.166667
[12] {tea=y}     => {bread=n}    0.3571429 0.8333333  1.458333
```

- if we are interested in rules with sugar as the conseqence

4. `rules_sugar<-apriori(d, parameter=list(supp=0.2, conf = 0.8, maxlen=3), appearance = list(default="lhs",rhs="sugar=y"))  # lhs - left hand sise, rhs right hand side of the rule`
5. `inspect(rules_sugar)`

```
> inspect(rules_sugar)
    lhs                     rhs        support   confidence lift
[1] {bread=y}            => {sugar=y} 0.3571429 0.8333333  1.666667
[2] {bread=y,tea=n}      => {sugar=y} 0.2857143 0.8000000  1.600000
[3] {bread=y,cheese=n}   => {sugar=y} 0.3571429 1.0000000  2.000000
[4] {bread=y,milk=n}     => {sugar=y} 0.3571429 0.8333333  1.666667
[5] {bread=y,butter=y}   => {sugar=y} 0.2857143 0.8000000  1.600000
[6] {cheese=n,honey=n}   => {sugar=y} 0.2142857 1.0000000  2.000000
[7] {butter=y,honey=n}   => {sugar=y} 0.2857143 0.8000000  1.600000
[8] {cheese=n,tea=n}     => {sugar=y} 0.2857143 0.8000000  1.600000
```