Data Mining

Association Analysis
Example Applications in which Co-Occurrence Matters

- We are often interested in co-occurrence relationships

  - **Marketing**
    1. identify items that are bought together by sufficiently many customers
    2. use this information for marketing or supermarket shelf management purposes

  - **Inventory Management**
    1. identify parts that are often needed together for repairs
    2. use this information to equip your repair vehicles with the right parts

  - **Usage Mining**
    1. identify words that frequently appear together in search queries
    2. use this information to offer auto-completion features to the user
Outline

1. Correlation Analysis
2. Association Analysis
   1. Frequent Itemset Generation
   2. Rule Generation
   3. Handling Continuous and Categorical Attributes
   4. Interestingness Measures
1. Correlation Analysis

- Correlation analysis measures the degree of dependency between two variables
  - Continuous variables: Pearson’s correlation coefficient (PCC)
  - Binary variables: Phi coefficient

\[
PCC(x, y) = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2} \sqrt{\sum (y_i - \bar{y})^2}}
\]

\[
Phi(x, y) = \frac{f_{11}f_{00} - f_{01}f_{10}}{\sqrt{f_{1+}f_{+1}f_{0+}f_{+0}}}
\]

- Value range [-1,1]
  - 1 : positive correlation
  - 0 : variables independent
  - -1 : negative correlation
Correlations between Products in Shopping Baskets

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basket 1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Basket 2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Basket 3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

1 : always bought together  
0 : sometimes bought together  
-1 : never bought together

Shortcoming: Measures correlation only between two items but not between multiple items, e.g. \{ThinkPad, Cover\} → \{Minimaus\}
2. Association Analysis

- Association analysis can find multiple item co-occurrence relationships (descriptive method)
- focuses on occurring items, not absent items
- first algorithms developed in the early 90s at IBM by Agrawal & Srikant
- initially used for **shopping basket analysis** to find how items purchased by customers are related
- later extended to more complex data structures
  - sequential patterns
  - subgraph patterns
- and other application domains
  - web usage mining, social science, life science
Association Analysis

Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction.

Shopping Transactions

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Milk</td>
</tr>
<tr>
<td>2</td>
<td>Bread, Diaper, Beer, Eggs</td>
</tr>
<tr>
<td>3</td>
<td>Milk, Diaper, Beer, Coke</td>
</tr>
<tr>
<td>4</td>
<td>Bread, Milk, Diaper, Beer</td>
</tr>
<tr>
<td>5</td>
<td>Bread, Milk, Diaper, Coke</td>
</tr>
</tbody>
</table>

Examples of Association Rules

\{Diaper\} \rightarrow \{Beer\}
\{Beer, Bread\} \rightarrow \{Milk\}
\{Milk, Bread\} \rightarrow \{Eggs, Coke\}

Implication means co-occurrence, not causality!
Definition: Support and Frequent Itemset

- **Itemset**
  - collection of one or more items
  - example: \{Milk, Bread, Diaper\}
  - \(k\)-itemset: An itemset that contains \(k\) items

- **Support count** \(\sigma\)
  - frequency of occurrence of an itemset
  - e.g. \(\sigma(\{\text{Milk, Bread, Diaper}\}) = 2\)

- **Support** \(s\)
  - fraction of transactions that contain an itemset
  - e.g. \(s(\{\text{Milk, Bread, Diaper}\}) = 2/5 = 0.4\)

- **Frequent Itemset**
  - an itemset whose support is greater than or equal to a minimal support \(\text{mins}u\text{p}\) threshold specified by the user

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Milk</td>
</tr>
<tr>
<td>2</td>
<td>Bread, Diaper, Beer, Eggs</td>
</tr>
<tr>
<td>3</td>
<td>Milk, Diaper, Beer, Coke</td>
</tr>
<tr>
<td>4</td>
<td>Bread, Milk, Diaper, Beer</td>
</tr>
<tr>
<td>5</td>
<td>Bread, Milk, Diaper, Coke</td>
</tr>
</tbody>
</table>
Definition: Association Rule

- **Association Rule**
  - an implication expression of the form \( X \rightarrow Y \), where \( X \) and \( Y \) are itemsets
  - an association rule states that when \( X \) occurs, \( Y \) occurs with certain probability.
  - Example:
    \[
    \{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\}
    \]
    Condition  Consequent

- **Rule Evaluation Metrics**
  - **Support** \( (s) \)
    fraction of transactions that contain both \( X \) and \( Y \)
  - **Confidence** \( (c) \)
    measures how often items in \( Y \) appear in transactions that contain \( X \)

\[
\begin{align*}
  s(X \rightarrow Y) &= \frac{|X \cup Y|}{|T|} \\
  c(X \rightarrow Y) &= \frac{\sigma(X \cup Y)}{\sigma(X)}
\end{align*}
\]

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Milk</td>
</tr>
<tr>
<td>2</td>
<td>Bread, Diaper, Beer, Eggs</td>
</tr>
<tr>
<td>3</td>
<td>Milk, Diaper, Beer, Coke</td>
</tr>
<tr>
<td>4</td>
<td>Bread, Milk, Diaper, Beer</td>
</tr>
<tr>
<td>5</td>
<td>Bread, Milk, Diaper, Coke</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
  s &= \frac{\sigma(\text{Milk, Diaper, Beer})}{|T|} = \frac{2}{5} = 0.4 \\
  c &= \frac{\sigma(\text{Milk, Diaper, Beer})}{\sigma(\text{Milk, Diaper, Beer})} = \frac{2}{3} = 0.67
\end{align*}
\]
Main Challenges concerning Association Analysis

1. Mining associations from large amounts of data can be computationally expensive
   • algorithms need to apply smart pruning strategies

2. Algorithms often discover a large number of associations
   • many of them are uninteresting or redundant
   • the user needs to select the subset of the associations that is relevant given her task at hand
The Association Rule Mining Task

- Given a set of transactions $T$, the goal of association rule mining is to find all rules having
  1. support $\geq \text{minsup}$ threshold
  2. confidence $\geq \text{minconf}$ threshold

- $\text{minsup}$ and $\text{minconf}$ are provided by the user.

- Brute Force Approach:
  1. list all possible association rules
  2. compute the support and confidence for each rule
  3. remove rules that fail the $\text{minsup}$ and $\text{minconf}$ thresholds

$\implies$ Computationally prohibitive due to large number of candidates!
Mining Association Rules

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Milk</td>
</tr>
<tr>
<td>2</td>
<td>Bread, Diaper, Beer, Eggs</td>
</tr>
<tr>
<td>3</td>
<td>Milk, Diaper, Beer, Coke</td>
</tr>
<tr>
<td>4</td>
<td>Bread, Milk, Diaper, Beer</td>
</tr>
<tr>
<td>5</td>
<td>Bread, Milk, Diaper, Coke</td>
</tr>
</tbody>
</table>

Example rules:

- \{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\} (s=0.4, c=0.67)
- \{\text{Milk, Beer}\} \rightarrow \{\text{Diaper}\} (s=0.4, c=1.0)
- \{\text{Diaper, Beer}\} \rightarrow \{\text{Milk}\} (s=0.4, c=0.67)
- \{\text{Beer}\} \rightarrow \{\text{Milk, Diaper}\} (s=0.4, c=0.67)
- \{\text{Diaper}\} \rightarrow \{\text{Milk, Beer}\} (s=0.4, c=0.5)
- \{\text{Milk}\} \rightarrow \{\text{Diaper, Beer}\} (s=0.4, c=0.5)

Observations:
- All the above rules are binary partitions of the same itemset: \{\text{Milk, Diaper, Beer}\}
- Rules originating from the same itemset have identical support but can have different confidence.
- Thus, we may decouple the support and confidence requirements.
Mining Association Rules

- Two-step approach:

  1. **Frequent Itemset Generation**
     - generate all itemsets whose support $\geq \minsup$

  2. **Rule Generation**
     - generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

- Frequent itemset generation is still computationally expensive
2.1 Frequent Itemset Generation

Given $d$ items, there are $2^d$ candidate itemsets!
Brute Force Approach

- Each itemset in the lattice is a candidate frequent itemset
- Count the support of each candidate by scanning the database
- Match each transaction against every candidate

Transactions

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Milk</td>
</tr>
<tr>
<td>2</td>
<td>Bread, Diaper, Beer, Eggs</td>
</tr>
<tr>
<td>3</td>
<td>Milk, Diaper, Beer, Coke</td>
</tr>
<tr>
<td>4</td>
<td>Bread, Milk, Diaper, Beer</td>
</tr>
<tr>
<td>5</td>
<td>Bread, Milk, Diaper, Coke</td>
</tr>
</tbody>
</table>

List of Candidates

- Complexity ~ O(NMw) ➔ Expensive since $M = 2^d$ !!!
- A smarter algorithm is required
Example: Brute Force Approach

- Example:
  - Amazon has 10 million books (i.e., Amazon Germany, as of 2011)
- That is $2^{10,000,000}$ possible itemsets
- As a number:
  - $9.04981... \times 10^{3,010,299}$
  - that is: a number with 3 million digits!
- However:
  - most itemsets will not be important at all, e.g., books on Chinese calligraphy, Inuit cooking, and data mining bought together
  - thus, smarter algorithms should be possible
  - intuition for the algorithm: All itemsets containing Inuit cooking are likely infrequent
Reducing the Number of Candidates

- **Apriori Principle**

  If an itemset is frequent, then all of its subsets must also be frequent.

- Apriori principle holds due to the following property of the support measure:

  \[ \forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y) \]

  - support of an itemset never exceeds the support of its subsets
  - this is known as the anti-monotone property of support
Using the Apriori Principle for Pruning

If an itemset is infrequent, then all of its supersets must also be infrequent.

Found to be Infrequent

Pruned supersets
Example: Using the Apriori Principle for Pruning

No need to generate candidates involving Coke or Eggs

No need to generate candidate \{Milk, Diaper, Beer\} as count \{Milk, Beer\} = 2
The Apriori Algorithm

1. Let $k=1$
2. Generate frequent itemsets of length 1
3. Repeat until no new frequent itemsets are identified
   1. Generate length $(k+1)$ candidate itemsets from length $k$ frequent itemsets
   2. Prune candidate itemsets that can not be frequent because they contain subsets of length $k$ that are infrequent (Apriori Principle)
   3. Count the support of each candidate by scanning the DB
   4. Eliminate candidates that are infrequent, leaving only those that are frequent
Example: Apriori Algorithm

Dataset T

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>T100</td>
<td>1, 3, 4</td>
</tr>
<tr>
<td>T200</td>
<td>2, 3, 5</td>
</tr>
<tr>
<td>T300</td>
<td>1, 2, 3, 5</td>
</tr>
<tr>
<td>T400</td>
<td>2, 5</td>
</tr>
</tbody>
</table>

**1. scan T**

- **Cand\(_1\):** \{1\}:2, \{2\}:3, \{3\}:3, \{4\}:1, \{5\}:3
- **Fequ\(_1\):** \{1\}:2, \{2\}:3, \{3\}:3, \{5\}:3
- **Cand\(_2\):** \{1,2\}, \{1,3\}, \{1,5\}, \{2,3\}, \{2,5\}, \{3,5\}

**2. scan T**

- **Cand\(_2\):** \{1,2\}:1, \{1,3\}:2, \{1,5\}:1, \{2,3\}:2, \{2,5\}:3, \{3,5\}:2
- **Fequ\(_2\):** \{1,3\}:2, \{2,3\}:2, \{2,5\}:3, \{3,5\}:2
- **Cand\(_3\):** \{2, 3, 5\}

**3. scan T**

- **C\(_3\):** \{2, 3, 5\}:2
- **F\(_3\):** \{2, 3, 5\}
Frequent Itemset Generation in Rapidminer and Python

**FP-Growth**

Alternative frequent itemset generation algorithm which compresses data into tree structure in memory. Details Tan/Steinbach/Kumar: Chapter 4.6

**Python**

```python
from orangecontrib.associate.fpgrowth import frequent_itemsets

# Calculate frequent itemsets
itemsets = dict(frequent_itemsets(dataset.values, 0.2))
```
### Frequent Itemsets in Rapidminer

<table>
<thead>
<tr>
<th>Size</th>
<th>Support</th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.600</td>
<td>Asus EeePC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.500</td>
<td>LT Minimaus</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.500</td>
<td>2 GB DDR3 RAM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.500</td>
<td>Asus EeePC</td>
<td>2 GB DDR3 RAM</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.400</td>
<td>ThinkPad X220</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.400</td>
<td>Netbook-Schutzhülle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.400</td>
<td>Lenovo Tablet Sleeve</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.400</td>
<td>LT Laser Maus</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.400</td>
<td>Asus EeePC</td>
<td>LT Minimaus</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.400</td>
<td>Asus EeePC</td>
<td>Netbook-Schutzhülle</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.400</td>
<td>2 GB DDR3 RAM</td>
<td>Netbook-Schutzhülle</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.400</td>
<td>Asus EeePC</td>
<td>2 GB DDR3 RAM</td>
<td>Netbook-Schutzhülle</td>
</tr>
<tr>
<td>1</td>
<td>0.300</td>
<td>HP Laserjet P2055</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.300</td>
<td>HP CE50 Toner</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.300</td>
<td>LT Minimaus</td>
<td>2 GB DDR3 RAM</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.300</td>
<td>LT Minimaus</td>
<td>Netbook-Schutzhülle</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.300</td>
<td>ThinkPad X220</td>
<td>Lenovo Tablet Sleeve</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.300</td>
<td>HP Laserjet P2055</td>
<td>HP CE50 Toner</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.300</td>
<td>Asus EeePC</td>
<td>LT Minimaus</td>
<td>2 GB DDR3 RAM</td>
</tr>
</tbody>
</table>
Example Application of Frequent Itemsets

1. Take top-k frequent itemsets of size 2 containing item A

2. Rank second item according to
   - profit made by selling item
   - whether you want to reduce number of items B in stock
   - knowledge about customer preferences

3. Offer special price for combination with top-ranked second item
2.2 Rule Generation

- Given a frequent itemset \( L \), find all non-empty subsets \( f \subseteq L \) such that \( f \rightarrow L - f \) satisfies the minimum confidence requirement.

Example Frequent Itemset:
{Milk, Diaper, Beer}

Example Rule:
{Milk, Diaper} \( \Rightarrow \) Beer

\[
c = \frac{\sigma(\text{Milk, Diaper, Beer})}{\sigma(\text{Milk, Diaper})} = \frac{2}{3} = 0.67
\]

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Milk</td>
</tr>
<tr>
<td>2</td>
<td>Bread, Diaper, Beer, Eggs</td>
</tr>
<tr>
<td>3</td>
<td>Milk, Diaper, Beer, Coke</td>
</tr>
<tr>
<td>4</td>
<td>Bread, Milk, Diaper, Beer</td>
</tr>
<tr>
<td>5</td>
<td>Bread, Milk, Diaper, Coke</td>
</tr>
</tbody>
</table>
Challenge: Large Number of Candidate Rules

- If \{A, B, C, D\} is a frequent itemset, then the candidate rules are:

  \[
  \begin{align*}
  &ABC \rightarrow D, &ABD \rightarrow C, &ACD \rightarrow B, &BCD \rightarrow A, \\
  &A \rightarrow BCD, &B \rightarrow ACD, &C \rightarrow ABD, &D \rightarrow ABC, \\
  &AB \rightarrow CD, &AC \rightarrow BD, &AD \rightarrow BC, &BC \rightarrow AD, \\
  &BD \rightarrow AC, &CD \rightarrow AB
  \end{align*}
  \]

- If \(|L| = k\), then there are \(2^k - 2\) candidate association rules (ignoring \(L \rightarrow \emptyset\) and \(\emptyset \rightarrow L\))
Rule Generation

How to efficiently generate rules from frequent itemsets?

• In general, confidence does not have an anti-monotone property
  \( c(ABC \rightarrow D) \) can be larger or smaller than \( c(AB \rightarrow D) \)

• But confidence of rules generated from the same itemset has an anti-monotone property

• e.g., \( L = \{A,B,C,D\} \):
  \[ c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD) \]

• Confidence is anti-monotone with respect to the number of items on the right hand side of the rule
Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

- i.e., “moving elements from left to right” cannot increase confidence

Reason:

\[ c(AB \rightarrow C) := \frac{s(ABC)}{s(AB)} \quad c(A \rightarrow BC) := \frac{s(ABC)}{s(A)} \]

- Due to anti-monotone property of support, we know
  \[ s(AB) \leq s(A) \]
  - Hence
  \[ c(AB \rightarrow C) \geq c(A \rightarrow BC) \]
Candidate Rule Pruning

Moving elements from left to right cannot increase confidence

Low Confidence Rule

Pruned Rule Candidates
Candidate Rule Generation within Apriori Algorithm

- Candidate rule is generated by merging two rules that share the same prefix in the rule consequent (right hand side of rule)

1. join(CD $\rightarrow$ AB, BD $\rightarrow$ AC) would produce the candidate rule D $\rightarrow$ ABC

2. Prune rule D $\rightarrow$ ABC if one of its parent rules does not have high confidence (e.g. AD $\rightarrow$ BC)

- All the required information for confidence computation has already been recorded in itemset generation.
- Thus, there is no need to scan the transaction data $T$ any more
Creating Association Rules in RapidMiner and Python

**RapidMiner**

**Python**

```python
from orangecontrib.associate.fpgrowth import association_rules

# Calculate association rules from itemsets
rules = association_rules(itemsets, 0.70)
```
Exploring Association Rules in Rapidminer

<table>
<thead>
<tr>
<th>No.</th>
<th>Premises</th>
<th>Conclusion</th>
<th>Support</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>58859</td>
<td>age = young</td>
<td>class = &lt;=50K</td>
<td>0.072</td>
<td>1</td>
</tr>
<tr>
<td>58860</td>
<td>native-country = US, age = young</td>
<td>class = &lt;=50K</td>
<td>0.067</td>
<td>1</td>
</tr>
<tr>
<td>58861</td>
<td>race = White, age = young</td>
<td>class = &lt;=50K</td>
<td>0.063</td>
<td>1</td>
</tr>
<tr>
<td>58862</td>
<td>workclass = Private, age = young</td>
<td>class = &lt;=50K</td>
<td>0.057</td>
<td>1</td>
</tr>
<tr>
<td>58863</td>
<td>hours-per-week = full-time, age = young</td>
<td>class = &lt;=50K</td>
<td>0.044</td>
<td>1</td>
</tr>
<tr>
<td>58864</td>
<td>sex = Male, age = young</td>
<td>class = &lt;=50K</td>
<td>0.039</td>
<td>1</td>
</tr>
<tr>
<td>58865</td>
<td>education = School, age = young</td>
<td>class = &lt;=50K</td>
<td>0.050</td>
<td>1</td>
</tr>
<tr>
<td>58866</td>
<td>sex = Female, age = young</td>
<td>class = &lt;=50K</td>
<td>0.032</td>
<td>1</td>
</tr>
<tr>
<td>58867</td>
<td>native-country = US, race = White, age = young</td>
<td>class = &lt;=50K</td>
<td>0.060</td>
<td>1</td>
</tr>
<tr>
<td>58868</td>
<td>native-country = US, workclass = Private, age = young</td>
<td>class = &lt;=50K</td>
<td>0.053</td>
<td>1</td>
</tr>
<tr>
<td>58869</td>
<td>native-country = US, hours-per-week = full-time, age = young</td>
<td>class = &lt;=50K</td>
<td>0.041</td>
<td>1</td>
</tr>
<tr>
<td>58870</td>
<td>native-country = US, sex = Male, age = young</td>
<td>class = &lt;=50K</td>
<td>0.037</td>
<td>1</td>
</tr>
<tr>
<td>58871</td>
<td>native-country = US, education = School, age = young</td>
<td>class = &lt;=50K</td>
<td>0.047</td>
<td>1</td>
</tr>
<tr>
<td>58872</td>
<td>native-country = US, sex = Female, age = young</td>
<td>class = &lt;=50K</td>
<td>0.030</td>
<td>1</td>
</tr>
<tr>
<td>58873</td>
<td>race = White, workclass = Private, age = young</td>
<td>class = &lt;=50K</td>
<td>0.051</td>
<td>1</td>
</tr>
<tr>
<td>58874</td>
<td>race = White, hours-per-week = full-time, age = young</td>
<td>class = &lt;=50K</td>
<td>0.039</td>
<td>1</td>
</tr>
<tr>
<td>58875</td>
<td>race = White, sex = Male, age = young</td>
<td>class = &lt;=50K</td>
<td>0.035</td>
<td>1</td>
</tr>
</tbody>
</table>

Filter by conclusion
Filter by confidence
### 2.3 Handling Continuous and Categorical Attributes

- How to apply association analysis to attributes that are not asymmetric binary variables?

<table>
<thead>
<tr>
<th>Session Id</th>
<th>Country</th>
<th>Session Length (sec)</th>
<th>Number of Web Pages viewed</th>
<th>Gender</th>
<th>Browser Type</th>
<th>Buy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>USA</td>
<td>982</td>
<td>8</td>
<td>Male</td>
<td>Chrome</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>China</td>
<td>811</td>
<td>10</td>
<td>Female</td>
<td>Chrome</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>USA</td>
<td>2125</td>
<td>45</td>
<td>Female</td>
<td>Firefox</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Germany</td>
<td>596</td>
<td>4</td>
<td>Male</td>
<td>IE</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Australia</td>
<td>123</td>
<td>9</td>
<td>Male</td>
<td>Firefox</td>
<td>No</td>
</tr>
</tbody>
</table>

- Example Rule:

\[\{\text{Number of Pages} \in [5,10) \land (\text{Browser}=\text{Firefox})\} \rightarrow \{\text{Buy} = \text{No}\}\]
Handling Categorical Attributes

- Transform categorical attribute into asymmetric binary variables
- Introduce a new “item” for each distinct attribute-value pair
  - e.g. replace “Browser Type” attribute with
    - attribute: “Browser Type = Chrome”
    - attribute: “Browser Type = Firefox”
    - ...

- Issues
  1. What if attribute has many possible values?
     - many of the attribute values may have very low support
     - potential solution: aggregate low-support attribute values
  2. What if distribution of attribute values is highly skewed?
     - example: 95% of the visitors have Buy = No
     - most of the items will be associated with (Buy=No) item
     - potential solution: drop the highly frequent item
Handling Continuous Attributes

- Transform continuous attribute into binary variables using discretization
  - equal-width binning
  - equal-frequency binning

- Issue: Size of the discretization intervals affects support & confidence
  - If intervals are too small
    - itemsets may not have enough support
  - If intervals too large
    - rules may not have enough confidence
    - e.g. combination of different age groups compared to a specific age group

{Refund = No, (Income = $51,251)} → {Cheat = No}
{Refund = No, (60K ≤ Income ≤ 80K)} → {Cheat = No}
{Refund = No, (0K ≤ Income ≤ 1B)} → {Cheat = No}
Attribute Transformation in RapidMiner and Python

- Categorical attribute values to binary attributes

```python
from sklearn.preprocessing import OneHotEncoder

# Apply one-hot encoding
coder = OneHotEncoder(sparse=False)
onehot_data = encoder.fit_transform(dataset)
```

- Continuous attribute values to binary attributes

```python
from sklearn.preprocessing import KBinsDiscretizer

# Discretize and one-hot encode dataset
discretizer = KBinsDiscretizer(n_bins=5, encode='onehot', strategy='quantile')
discretized_onehot_data = discretizer.fit_transform(dataset)
```
2.4 Interestingness Measures

- Association rule algorithms tend to produce too many rules
  - many of them are uninteresting or redundant
  - redundant if \( \{A, B, C\} \rightarrow \{D\} \) and \( \{A, B\} \rightarrow \{D\} \)
    have same support & confidence

- Interestingness of patterns depends on application
  - one man's rubbish may be another's treasure

- Interestingness measures can be used to prune or rank the derived rules.

- In the original formulation of association rules, support & confidence were the only interestingness measures used.

- Later, various other measures have been proposed
  - See Tan/Steinbach/Kumar, Chapter 6.7
  - We will have a look at one: Lift
### Drawback of Confidence

**Contingency table**

<table>
<thead>
<tr>
<th></th>
<th>Coffee</th>
<th>Coffee</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tea</td>
<td>15</td>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>Tea</td>
<td>75</td>
<td>5</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>10</td>
<td>100</td>
</tr>
</tbody>
</table>

**Association Rule: Tea → Coffee**

- confidence(Tea → Coffee) = 0.75
- but support(Coffee) = 0.9
- although confidence is high, **rule is misleading** as the fraction of coffee drinkers is higher than the confidence of the rule
- we want confidence(X → Y) > support(Y)
- otherwise rule is misleading as X reduces probability of Y
Lift

- The lift of an association rule $X \rightarrow Y$ is defined as:

$$Lift = \frac{c(X \rightarrow Y)}{s(Y)}$$

- Confidence normalized by support of consequent

- Interpretation
  - if $lift > 1$, then $X$ and $Y$ are positively correlated
  - if $lift = 1$, then $X$ and $Y$ are independent
  - if $lift < 1$, then $X$ and $Y$ are negatively correlated
Example: Lift

Contingency table

<table>
<thead>
<tr>
<th></th>
<th>Coffee</th>
<th>Coffee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tea</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td>Tea</td>
<td>75</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>10</td>
</tr>
</tbody>
</table>

\[
Lift = \frac{c(X \rightarrow Y)}{s(Y)}
\]

Association Rule: Tea $\rightarrow$ Coffee

- confidence(Tea $\rightarrow$ Coffee) = 0.75
- but support(Coffee) = 0.9

\[
Lift(\text{Tea } \rightarrow \text{Coffee}) = \frac{0.75}{0.9} = 0.8333
\]

- lift < 1, therefore is negatively correlated
Exploring Association Rules in RapidMiner

Lift close to 1

Solid lift
Conclusion

- The algorithm does the counting for you and finds patterns in the data

- You need to do the interpretation based on your knowledge about the application domain.
  - Which patterns are meaningful?
  - Which patterns are surprising?
Literature for this Slideset


**Chapter 4: Association Analysis:**
Basic Concepts and Algorithms

**Chapter 7: Association Analysis:**
Advanced Concepts