Lecture outline

- What is association mining?
- Market basket analysis and association rule examples
- Basic concepts and formalism
- Basic rule measurements
- The Apriori algorithm
- Performance bottlenecks in Apriori
- Multi-level and multi-dimensional association mining
- Quantitative association mining
- Constraint based mining
- Visualising association rules

What is association mining?

- Association mining is the task of finding frequent rules / associations / patterns / correlations / causal structures within (large) sets of items in transactional (relational) databases
- Unsupervised learning techniques (descriptive data mining, not predictive data mining)
- The main applications are
  - Market basket analysis (customers who buy X also buy Y)
  - Web log analysis (click-stream)
  - Cross-marketing
  - Sale campaign analysis
  - DNS sequence analysis

Association rules examples

- Rules form: body \(\Rightarrow\) head [support, confidence]
- Market basket:
  \(\text{buys}(X, \text{beer}) \Rightarrow \text{buys}(X, \text{snacks})\) [1\%, 60\%]
  - If a customer X purchased 'beer', in 60\% she or he also purchased 'snacks'
  - 1\% of all transactions contain the items 'beer' and 'snacks'
- Student grades:
  \(\text{major}(X, \text{MComp}) \text{ and takes}(X, \text{COMP8400}) \Rightarrow \text{grade}(X, \text{D})\) [3\%, 60\%]
  - If a student X, who's degree is 'MComp', took the course 'COMP8400' she or he in 60\% achieved a grade 'D'
  - The combination 'MComp', 'COMP8400' and 'D' appears in 3\% of all transactions (records) in the database

Basic concepts

- Given:
  - A (large) database of transactions
  - Each transaction contains a list of one or more items (e.g., purchased by a customer in a visit)
  - Find the rules that correlate the presence of one set of items with that of another set of items
- Normally one is only interested in rules that are frequent
  - For example, 70\% of customers who buy tires and car accessories also get their car service done
Question: How can this be improved to 80\%? Possibly offer special deals like a 15\% reduction of tire costs when the service is done
Rule measurements example (2)

<table>
<thead>
<tr>
<th>Transaction ID</th>
<th>Items Bought</th>
<th>Itemset</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>a, b, c</td>
<td>a</td>
<td>75.00%</td>
</tr>
<tr>
<td>1000</td>
<td>a, c</td>
<td>b</td>
<td>50.00%</td>
</tr>
<tr>
<td>4000</td>
<td>a, d</td>
<td>c</td>
<td>50.00%</td>
</tr>
<tr>
<td>5000</td>
<td>b, d, e, f</td>
<td>a, c</td>
<td>50.00%</td>
</tr>
</tbody>
</table>

- Minimum support = 50% and confidence = 50%
- Rule a ⇒ c
  - support (a ⇒ c): 50%
  - confidence (a ⇒ c) = support(a ⇒ c) / support(a) = 50% / 75% = 66.67%

Mining frequent item sets

- Key step: Find the frequent sets of items that have minimum support (appear in at least 50% of all transactions in a database)
- Basic principle (Apriori principle): A sub-set of a frequent item set must also be a frequent item set
  - For example, if {a,b} is frequent, both {a} and {b} have to be frequent (if ‘beer’ and ‘chips’ are purchased frequently together, then ‘beer’ is purchased frequently and ‘chips’ are also purchased frequently)
- Basic approach: Iteratively find frequent item sets with cardinality from 1 to k (k-item sets), k > 1
- Use the frequent item sets to generate association rules
  - For example, frequent 3-item set {a,b,c} contains rules:
    - a ⇒ {b, c}
    - a ⇒ b, (a,b) ⇒ c
    - a, b ⇒ c
    - a, b, c ⇒ d
    - a, c ⇒ b
    - a, d ⇒ c
    - a, e ⇒ f

  - We are normally only interested in longer rules (with all except one element on the left-hand side)

The Apriori algorithm (Agrawal & Srikant, VLDB’94)

- $C_k$: Candidate item set of size k
- $L_k$: Frequent item set of size k
- Pseudo-code:
  
  
  ```
  L_1 = (frequent items);
  for (k = 1; L_k ≠ ∅; k++) do
  
  C_k = candidates generated from $L_{k-1}$;
  for each transaction t in database do
    increment the count of all candidates in $C_k$ that are contained in t
  $L_k$ = candidates in $C_k$ with min_support
  end do
  return $L_k$
  ```
The Apriori algorithm – An example (sup=50%)

Database D

<table>
<thead>
<tr>
<th>ID</th>
<th>Items</th>
<th>Scan D</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>a,b,c,e</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>a,b,c,e</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>a,b,c,e</td>
<td></td>
</tr>
<tr>
<td>400</td>
<td>b,c,e</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L_1</th>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>(e)</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L_2</th>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a,b)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>(a,c)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>(a,e)</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>(b,c)</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>(b,e)</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>(c,e)</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>L_3</th>
<th>Itemset</th>
<th>Sup</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a,b,c)</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

How to generate candidate item-sets?
- Suppose the items in L_{k-1} are listed in an order (e.g., a < b)
- **Step 1:** Self-joining L_{k-1}
  - Insert into C_k
    - Select p.item_1, p.item_2, ..., p.item_{k-2}, q.item_{k-1} from L_{k-1} and q,
    - Where p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2}

- **Step 2:** Pruning
  - For all (k-1)-sub-sets s of c do
    - If s is not in L_{k-1} then delete c from C_k

Apriori performance bottlenecks
- The core of the Apriori algorithm is to
  - Use frequent (k-1)-item sets to generate candidate frequent k-item sets
  - Use database scan and pattern matching to collect counts for candidate item sets
- **Candidate generation is the main bottleneck**
  - 10^k frequent 1-item sets (sets of length 1) will generate 10^k candidate 2-item sets
  - To discover a frequent pattern of size 100 (for example \(\{a_1, a_2, ..., a_{100}\}\))
    one needs to generate \(2^{100} = 10^{30}\) candidates
  - Multiple scans of the database are needed (n+1 scans if the longest pattern is n items long)

Methods to improve Apriori’s efficiency
- **Reduce the number of scans of the database**
  - Any item set that is potentially frequent in the database must be frequent in at least one of the partitions of the database
  - **Scan 1:** Partition database and find local frequent patterns
  - **Scan 2:** Consolidate global frequent patterns
- **Shrink number of candidates**
  - Select a sample of the database, mine frequent patterns within sample using Apriori
  - **Scan 3:** Scan database once to verify frequent item sets found in sample
  - **Scan 4:** Scan database again to find missed frequent patterns
- **Facilitate support of counting candidates**
  - For example, use special data structures like Frequent-Pattern tree (FP-tree)
Multi-level association mining

- Items often form hierarchies
- Items at lower levels are expected to have lower support
  - Flexible support setting (uniform, reduced, or group-based (user specific))

Multi-level association mining (2)

- Some rules may be redundant due to ancestor relationships between items
- For example:
  
  \[ \text{buys}(X, \text{'milk'}) \Rightarrow \text{buys}(X, \text{'bread'}) \ [8\%, 70\%] \]
  \[ \text{buys}(X, \text{'skim milk'}) \Rightarrow \text{buys}(X, \text{'bread'}) \ [2\%, 72\%] \]

  - The first rule is said to be an ancestor of the second rule
  - A rule is redundant if its support is close to the "expected" value, based on the rule’s ancestor
    
    - For example, if around 25% of all milk purchased is ‘skim milk’, then the second rule above is redundant, as it has a ¼ of the support of the first, more general rule (and similar confidence)

Multi-dimensional association mining

- Single-dimensional rules: \( \text{buys}(X, \text{'milk'}) \Rightarrow \text{buys}(X, \text{'bread'}) \)
- Multi-dimensional rules: Two or more dimensions or predicates (or attributes)
  - Inter-dimension association rules (no repeated predicates): \( \text{age}(X, '19-25') \) and \( \text{occupation}(X, 'student') \Rightarrow \text{buys}(X, \text{'coke'}) \)
  - Hybrid-dimension association rules (repeated predicates): \( \text{age}(X, '19-25') \) and \( \text{buys}(X, \text{'popcorn'}) \Rightarrow \text{buys}(X, \text{'coke'}) \)

Categorical Attributes: finite number of possible values, no ordering among values (data cube approach)
Quantitative Attributes: numeric, implicit ordering among values (discretisation, clustering, etc.)

Quantitative association mining

- Techniques can be categorised by how numerical attributes, such as age or income, are treated
- Static discretisation based on predefined concept hierarchies
- Dynamic discretisation based on data distribution
  \( A_{\text{new}} \) and \( A_{\text{new}} = A_{\text{old}} \)
  - Example: \( \text{age}(X, '19-25') \) and \( \text{income}(X, '40K-60K') \Rightarrow \text{buys}(X, \text{'HDTV'}) \)

- For quantitative rules, do discretisation such that (for example) the confidence of the rules mined is maximised

Mining interesting correlation patterns

- Flexible support
  - Some items might be very rare but are valuable (like diamonds)
  - Customise support specification and application
- Top-k frequent patterns
  - It can be hard to specify support, but top-k rules with length are more desirable
  - Achievable using special data structures, like Frequent Pattern (FP) tree
  - Dynamically raise support during FP-tree construction phase, and select most promising to mine

Constraint based data mining

- Finding all the frequent rules or patterns in a database autonomously is unrealistic
  - The rules / patterns could be too many and not focussed
- Data mining should be an interactive process
  - The user directs what should be mined using a data mining query language or a graphical user interface
- Constraint-based mining
  - User flexibility: provides constraints on what to be mined (and what not)
  - System optimisation: explores such constraints for efficient mining
Constraints in data mining

- Knowledge type constraint
  - Correlation, association, etc.
- Data constraint (use SQL like queries)
  - For example: Find product pairs sold frequently in both stores in Sydney and Melbourne
- Dimension / level constraint
  - In relevance to region, price, brand, customer category, etc.
- Rule or pattern constraint
  - Small sales (price < $10) trigger big sales (sum > $200)
- Interestingness constraint
  - Strong rules only: support_{min} > 3%, confidence_{min} > 75%