Chapter 4 - Data Mining
A Short Introduction
Today's Question

1. Data Mining Overview
2. Association Rule Mining
3. Clustering
4. Classification

Interpretations and their Evaluation

- **The "database approach"**
  - consult the user in an application
  - develop a conceptual model
  - develop, implement and use the logical model
  - re-consult the user and start over

- **The "data mining approach"**
  - take a learning dataset
  - build model from it
  - take a test dataset
  - compute how well the model matches

- **The "information retrieval approach"**
  - ask human users for the relevance of information for a profile
  - apply the retrieval algorithm to the same problem
  - compare the results: recall, precision

Example Data Mining
1. Data Mining Overview

- Data acquisition and data storage result in huge databases
  - Supermarket and credit card transactions (market analysis)
  - Scientific data
  - Web analysis (browsing behavior, advanced information retrieval)

- Definition

  Data mining is the analysis of large observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful for the data owner.

- Extraction of information from data

The wide-spread use of distributed information systems leads to the construction of large data collections in business, science and on the Web. These data collections contain a wealth of information, that however needs to be discovered. Businesses can learn from their transaction data more about the behavior of their customers and therefore can improve their business by exploiting this knowledge. Science can obtain from observational data (e.g. satellite data) new insights on research questions. Web usage information can be analyzed and exploited to optimize information access.

Data mining provides methods that allow to extract from large data collections unknown relationships among the data items that are useful for decision making. Thus data mining generates novel, unsuspected interpretations of data.
The Classical Example: Association Rule Mining

- **Market basket analysis**
  - Given a database of purchase transactions and for each transaction a list of purchased items
  - Find rules that correlate a set of items occurring in a list with another set of items

- **Example**
  - 98% of people who purchase tires and auto accessories also get automotive services done
  - 60% of people who buy diapers also buy a beer
  - 90% of people who buy Neal Stephenson's "Snow Crash" at amazon also buy "Cryptonomicon"
  - etc.

The classical example of a data mining problem is "market basket analysis". Stores gather information on what items are purchased by their customers. The hope is, by finding out what products are frequently purchased jointly (i.e. are associated with each other), being able to optimize the marketing of the products (e.g. the layout of the store) by better targeting certain groups of customers. A famous example was the discovery that people who buy diapers also frequently buy beers (probably exhausted fathers of small children). Therefore nowadays one finds frequently beer close to diapers (and of course also chips close to beer) in supermarkets. Similarly, amazon exploits this type of associations in order to propose to their customers books that are likely to match their interests.

This problem was the starting point for one of the best known data mining techniques: association rule mining.
The task of discovering interesting patterns is part of a larger process supported by data mining systems.

- For being able to mine data, first the data needs to be collected from the available data sources. Since these data sources can be distributed and heterogeneous databases, database integration techniques need to be applied. The integrated data is kept in so-called data warehouses, databases replicating and consolidating the data from the various data sources. An important concern when integrating the data sources is data cleansing, i.e. removing inconsistent and faulty data as far as possible. The tasks of data integration and data cleansing are supported by so-called data warehousing systems.

- Once the data is consolidated in a data warehouse, for specific data mining tasks, i.e. tasks having a specific purpose in mind, the data can be selected from the data warehouse. This task-specific data collections are often called data-marts. The data-mart is the database to which the specific data mining task, i.e. the discovery process, is applied.

- The data mining task is the process of detecting interesting patterns in the data. This is what generally is understood as data mining in the narrow sense. We will introduce in the course examples of techniques that are used to perform this task (e.g. association rule mining).

- Once specific patterns are detected they can be further processed. Further processing can imply the evaluation of their "interestingness" for the specific problem studied and the implementation of certain actions to react on the discovery of the pattern.

Each of the steps described can influence the preceding steps. For example, patterns or outliers detected during data mining may indicate the presence of erroneous data rather than of interesting features in the source databases. This may imply adaptations of the data cleansing during data integration.
Data mining techniques can be classified according to the goals they pursue and the results they provide.

A basic distinction is made among techniques that provide a global statement on the data, in the form of summaries and globally applicable rules, or that provide local statements only, in the form of rare patterns or exceptional dependencies in the data.

The example of association rule mining is a typical case of discovering local patterns. The rules obtained are unexpected (unprobable) patterns and typically relate to small parts of the database only.

Techniques for providing global statements on the data are further distinguished into techniques that are used to "simply" describe the data and into techniques that allow to make predictions on data if partial data is known. Descriptive modeling techniques provide compact summaries of the databases, for example, by identifying clusters of similar data items. Predictive modeling techniques provide globally applicable rules for the database, for example, allowing to predict properties of data items, if some of their properties are known.

Information retrieval is usually also considered as a special case of data mining, where given patterns are searched for in the database. Finally, exploratory data analysis is used when no clear idea exists, what is being sought for in the database. It may serve as a preprocessing step for more specific data mining tasks.
Components of a Data Mining Algorithm

- **Model or pattern structure**
  - Which kind of global model or local pattern is searched
  - *Vector representation of documents*

- **Score function**
  - Determine how well a given data set fits a model or pattern
  - *Similarity of vectors*

- **Optimization and search method**
  - Finding best parameters for a global model: optimization problem
  - Finding data satisfying a pattern: search problem
  - *Search the k nearest neighbors*

- **Data management strategy**
  - Handling very large datasets
  - *Inverted files*

Each data mining method can be characterized in terms of four aspects:

- The models or patterns that are used to describe what is searched for in the data set. Typical models are dependency rules, clusters and decision trees.

- The scoring functions that are used to determine how well a given dataset fits the model. This is comparable to the similarity functions used in information retrieval.

- The method that is applied in order to find data in the dataset that scores well with respect to the scoring function. Normally this requires efficient search algorithms that allow to identify those models that fit the data well according to the scoring functions.

- Finally the scalable implementation of the method for large datasets. Here indexing techniques and efficient secondary storage management are applied.

In particular the last two issues differentiate data mining from related areas like statistics and machine learning: scalability for large databases is a key problem in data mining and only statistical and machine learning techniques that scale well are applicable for data mining.

For illustration we identify the components of information retrieval, when looked at as data mining method.
Summary

• What is the purpose of data mining?
• Which preprocessing steps are required for a data mining task?
• Which are the four aspects that characterize a data mining method?
• What is the difference between classification and prediction?
• Explain of how information retrieval can be characterized as a data mining method?
2. Association Rule Mining

- **Search patterns given as association rules of the form**

  \[ \text{Body} \Rightarrow \text{Head} \ [\text{support}, \text{confidence}] \]

  **Body**: property of an object \(x\) e.g. a transaction, a person
  **Head**: property probable to be implied by Body
  **support, confidence**: measures on validity of the rule

- **Examples**
  - \(\text{buys}(x, \text{"diapers"}) \Rightarrow \text{buys}(x, \text{"beers"}) \ [0.5\%, \ 60\%]\)
  - \(\text{major}(x, \text{"CS"}) \land \text{takes}(x, \text{"DB"}) \Rightarrow \text{grade}(x, \text{"A"}) \ [1\%, \ 75\%]\)

- **Problem**: Given
  1. database of transactions
  2. each transaction is a list of items

  **Find**: all rules that correlate the presence of one set of items with that of another set of items

Association rule mining is a technique for discovering unsuspected data dependencies and is one of the best known data mining techniques. The basic idea is to identify from a given database, consisting of itemsets (e.g. shopping baskets), whether the occurrence of specific items, implies also the occurrence of other items with a relatively high probability. In principle the answer to this question could be easily found by exhaustive exploration of all possible dependencies, which is however prohibitively expensive. Association rule mining thus solves the problem of how to search efficiently for those dependencies.
In the "logical" notation we have used before in order to express association rules, it was possible to establish dependencies among different types of predicates applied to the items. These general types of rules are called multi-dimensional association rules. However, it is straightforward to transform multi-dimensional association rules into single-dimensional rules, by considering different predicates applied to the same items as different items. Therefore in the following we will only consider single-dimensional association rules.
Support and Confidence

Let minimum support 50%
and minimum confidence 50%

buys(x,"beer") ⇒ buys(x,"diaper") [50%, 66.6%]
buys(x,"diaper") ⇒ buys(x,"beer") [50%, 100%]

This example illustrates the basic notions used in association rule mining: transactions, itemsets, support and confidence. Transaction consist of a transaction identifier and an itemset. The itemset is the set of items that occur jointly in a transactions (e.g. the items bought). Support is the number of transactions in which the association rule holds, i.e. in which all items of the rule occur (e.g. both beer and diaper). If this number is too small, probably the rule is not important or accidentally true. Confidence is the probability that in case the head of the rule (the condition) is satisfied also the body of the rule (the conclusion) is satisfied. This indicates to which degree the rule is actually true, in the cases where the rule is applicable.
Support and Confidence: Possible Situations

- Assume support for $A \cup B$ is high (above threshold)

![Diagram showing possible situations](image)

This figure illustrates the meaning and importance of the "directionality" of association rules. We assume that in all cases the intersection areas (i.e. the support) are above the required threshold. Then four cases are possible as shown. Thus association rules not only express a high probability of co-occurrence of items, such as in the last case, but also conditional dependencies among the occurrences of items (or inclusion relationships).
### Definition of Association Rules

**Terminology and Notation**

- **Set of all items** $I$, subset of $I$ is called **itemset**
- **Transaction** $(tid, T)$, $T \subseteq I$ itemset, transaction identifier $tid$
- **Set of all transactions** $D$ (database), $Transaction T \in D$

**Definition of Association Rules** $A \Rightarrow B \ [s, c]$

- $A, B$ itemsets $(A, B \subseteq I)$
- $A \cap B$ empty
- **support** $s$ = probability that a transaction contains $A \cup B$
  \[ s = P(A \cup B) \]
- **confidence** $c$ = conditional probability that a transaction having $A$ also contains $B$
  \[ c = P(B|A) \]

**Example:**

- **Items** $I = \{apple, beer, diaper, eggs, milk\}$
- **Transaction** $(2000, \{beer, diaper, milk\})$
- **Association rule** $(beer) \Rightarrow \{diaper\} \ [0.5, 0.66]$

This is a summary of the basic notations and notions used in association rule mining.
Here we summarize the most important ideas that will allow to search for association rules efficiently.

First, a necessary condition for finding an association rule of form A→B is sufficiently high support. Therefore, for finding such rules, we have first to find itemsets within the transactions that occur sufficiently frequent. These are called frequent itemsets.

Second we can observe that any subset of a frequent itemset is necessarily also a frequent itemset (this is called the apriori property).

Third, we can exploit this observation in order to reduce the number of itemsets that need to be considered in the search. Once frequent itemsets of lower cardinality are found, only itemsets of larger cardinality need to be considered that contain one of the frequent itemsets already found. This allows to reduce the search space drastically as we will see.
Exploiting the Apriori Property (1)

• If we know the frequent (k-1)-itemsets, which are candidates for being frequent k-itemsets?

- if we know all frequent (k-1)-itemsets \( L_{k-1} \), then we can construct a candidate set \( C_k \) for frequent k-itemsets by joining two frequent (k-1)-itemsets that differ by exactly 1 item: join step
  - only these itemsets CAN BE frequent k-itemsets

Assume that we know frequent itemsets of size k-1. Considering a k-itemset we can immediately conclude that by dropping two different items we have two frequent (k-1) itemsets. From another perspective this can be seen as a possible way to construct k-itemsets. We take two (k-1) item sets which differ only by one item and take their union. This step is called the join step and is used to construct POTENTIAL frequent k-itemsets.
Algorithm for Creating Candidates

- Suppose the items in $L_{k-1}$ are increasingly sorted in a list, then $C_k$ is created through the following SQL statement

$$\text{insert into } C_k$$

$$\text{SELECT \ p.item_1, \ p.item_2, \ \ldots, \ p.item_{k-1}, \ q.item_{k-1}}$$

$$\text{FROM } L_{k-1} \ p, \ L_{k-1} \ q$$

$$\text{WHERE p.item_1=q.item_1, \ \ldots, \ p.item_{k-2}=q.item_{k-2}, \ p.item_{k-1} < q.item_{k-1}}$$

<table>
<thead>
<tr>
<th>item1</th>
<th>item2</th>
<th>...</th>
<th>itemk-2</th>
<th>itemk-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,1</td>
<td>2,1</td>
<td>...</td>
<td>k-2,1</td>
<td>k-1,1</td>
</tr>
<tr>
<td>1,1</td>
<td>2,1</td>
<td></td>
<td>k-2,1</td>
<td>k-1,2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1,1</td>
<td>2,1</td>
<td></td>
<td>k-2,1</td>
<td>k-1,p1</td>
</tr>
<tr>
<td>1,2</td>
<td>2,2</td>
<td></td>
<td>k-2,2</td>
<td>k-1,p1+1</td>
</tr>
<tr>
<td>1,2</td>
<td>2,2</td>
<td></td>
<td>k-2,2</td>
<td>k-1,p1+2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1,2</td>
<td>2,2</td>
<td></td>
<td>k-2,2</td>
<td>k-1,p2</td>
</tr>
</tbody>
</table>

We may express the step of creating all possible combinations of (k-1) itemsets by the SQL query shown above, assuming the itemsets are stored as relations with attributes item1, ... , itemk. The table illustrates how these combinations are obtained. For each subset of items that share the first k-2 items, we construct all possible k-itemsets, by taking all possible, ordered pairs from the last column.
Exploiting the Apriori Property (2)

- A candidate itemset still not necessarily satisfies the apriori property

<table>
<thead>
<tr>
<th>Itemset</th>
<th>C3</th>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1 2}</td>
<td></td>
<td>{1 2 3}</td>
</tr>
<tr>
<td>{1 3}</td>
<td></td>
<td>{1 2 5}</td>
</tr>
<tr>
<td>{1 5}</td>
<td></td>
<td>{1 3 5}</td>
</tr>
<tr>
<td>{2 3}</td>
<td></td>
<td>{2 3 5}</td>
</tr>
<tr>
<td>{2 4}</td>
<td></td>
<td>{2 3 4}</td>
</tr>
<tr>
<td>{2 5}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- After generating the candidate set $C_k$, eliminate all itemsets for which not ALL (k-1)-itemsets are elements of $L_{k-1}$, i.e., are frequent (k-1) itemsets: prune step

- Only then count the remaining candidate k-itemsets in the database and eliminate those that are not frequent (sufficient condition)

The k-itemsets constructed in the join step not necessarily are frequent k-itemsets. One possible reason is that they contain some subset of items which is not frequent. These are eliminated in a prune step, by considering all itemsets of lower cardinality that have been constructed earlier.

After that, it is still possible that among the remaining k-itemsets some are not frequent when determining their frequency in the database. The important point is that this last check, which is expensive as it requires access to the complete database, needs to be performed for much fewer itemsets, since many possibilities have been eliminated in the join and prune step.
Generating Frequent Itemsets: The Apriori Algorithm

\[ k:=1; \quad L_k := \{ \text{frequent items in D} \}; \]

\[ \textbf{while } L_k \neq \emptyset \{
\]

\[ C_{k+1} := \text{candidates generated from } L_k \text{ by joining and pruning;} \]

\[ \textbf{for each} \quad \text{transaction } T \text{ in the database} \]

\[ \text{increment the count of all candidate item sets in } C_{k+1} \]

\[ \text{that are contained in } T; \]

\[ L_{k+1} := \text{candidates in } C_{k+1} \text{ with min\_support;} \]

\[ k := k+1; \} \]

\[ \textbf{return } \bigcup_k L_k; \]

This is the complete apriori algorithm for determining frequent itemsets.
Notice in this example of how the scan steps (when determining the frequency with respect to the database) eliminates certain items. Notice that in this example pruning does not apply.
Generating Association Rules from Frequent Itemsets

For each frequent itemset $L$ generate all non-empty subsets $S$

For every nonempty subset $S$ output the rule $S \Rightarrow L \setminus S$ if

$$\frac{sc(L)}{sc(S)} \geq \text{min}\_\text{conf}$$

$sc =$ support count,  
$min\_\text{conf} =$ minimal confidence

since

$$\text{confidence}(A \Rightarrow B) = P(B | A) = \frac{sc(A \cup B)}{sc(A)} \geq \text{min}\_\text{conf}$$

Once the frequent itemsets are found the derivation of association rules is straightforward: one checks for every frequent itemset whether there exists a subset $S$ that can occur as the head of a rule. For doing that, the support count, i.e. the frequency of the itemset in the database, which was obtained during the execution of the apriori algorithm, is used to compute the confidence (as a conditional probability). Note that also $L \setminus S$ is a frequent itemset, and therefore the support count is available for that set from the apriori algorithm.
Example

- Assume minimal confidence (min_conf) = 0.75

L = \{2, 3, 5\}, S = \{3, 5\}, confidence(S \Rightarrow L \setminus S) = \frac{\text{sc}(L)}{\text{sc}(S)} = \frac{2}{2}
therefore \{3, 5\} \Rightarrow \{2\} ok

L = \{2, 3, 5\}, S = \{2\}, confidence(S \Rightarrow L \setminus S) = \frac{\text{sc}(L)}{\text{sc}(S)} = \frac{2}{3}
therefore \{2\} \Rightarrow \{3, 5\} not ok

\[
\begin{array}{|c|c|c|}
\hline
\text{TID} & \text{Items} & \text{itemset} & \text{sup} \\
\hline
100 & 1 3 4 & \{1 \ 3\} & 2 \\
200 & 2 3 5 & \{2 \ 3\} & 2 \\
300 & 1 2 3 5 & \{2 \ 5\} & 3 \\
400 & 2 5 & \{3 \ 5\} & 2 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
\text{itemset} & \text{sup.} \\
\hline
\{1\} & 2 \\
\{2\} & 3 \\
\{3\} & 3 \\
\{5\} & 3 \\
\{2 \ 3 \ 5\} & 2 \\
\hline
\end{array}
\]

frequent itemsets
Improving Apriori’s Efficiency

- Transaction reduction
  - A transaction that does not contain any frequent k-itemset is useless in subsequent scans

- Partitioning
  - Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB

- Sampling
  - mining on a subset of given data, lower support threshold + a method to determine the completeness

- Many further advanced techniques

Though the basic apriori algorithm is designed to work efficiently for large datasets, there exist a number of possible improvements:

- Transactions in the database that turn out to contain no frequent k-itemsets can be omitted in subsequent database scans.

- One can try to identify first frequent itemsets in partitions of the database. This methods is based on the assumption that if an itemset is not frequent in one of the partitions at least (local frequent itemset) then it will also not be frequent in the whole database.

- The sampling method selects samples from the database and searches for frequent itemsets in the sampled database using a correspondingly lower threshold for the support.
Mining Multidimensional Association Rules

- Single-dimensional rules
  \[ \text{buys}(X, \text{"milk"}) \Rightarrow \text{buys}(X, \text{"bread"}) \]

- Multi-dimensional rules: more than 2 dimensions or predicates
  - Inter-dimension association rules (no repeated predicates)
    \[ \text{age}(X, \text{"19-25"}) \land \text{occupation}(X, \text{"student"}) \Rightarrow \text{buys}(X, \text{"coke"}) \]
  - Hybrid-dimension association rules (repeated predicates)
    \[ \text{age}(X, \text{"19-25"}) \land \text{buys}(X, \text{"popcorn"}) \Rightarrow \text{buys}(X, \text{"coke"}) \]

- Transformation into single-dimensional rules
  - Items are predicate/value pairs
    \[ \text{customer}(X, \text{[age, "19-25"]}) \land \text{customer}(X, \text{[occupation, "student"]}) \Rightarrow \text{customer}(X, \text{[buys, "coke"]}) \]
    \[ \text{customer}(X, \text{[age, "19-25"]}) \land \text{customer}(X, \text{[buys, "popcorn"]}) \Rightarrow \text{customer}(X, \text{[buys, "coke"]}) \]

Multidimensional association rules can be mined using the same method by transforming the problem. The items and the corresponding item values are encoded into a tuple. This results again in a finite number of possible (modified) item values, and therefore the same techniques as for single-dimensional rules apply.
Mining Quantitative Association Rules

- Categorical Attributes
  - finite number of possible values, no ordering among values
- Quantitative Attributes
  - numeric, implicit ordering among values

- Quantitative attributes are transformed into categorical attributes by
  - Static discretization of quantitative attributes
    - Quantitative attributes are statically discretized by using predefined concept hierarchies.
  - Dynamic discretization
    - Quantitative attributes are dynamically discretized into "bins" based on the distribution of the data.

For quantitative attributes the situation is more complex. A simple approach is to statically or dynamically discretize them into categorical attributes.

However, the rules that can be found depend on the discretization chosen. It may happen that the bins are for example too fine-grained, and a rule that could be more efficiently be expressed at a coarser granularity is split into multiple rules.

For example: if age is discretized into steps of 2 years we would probably find rules
Age(X, 18..19) and lives(X, Lausanne) -> profession(X, student)
Age(X, 20..21) and lives(X, Lausanne) -> profession(X, student)
Could be also expressed as a rule
Age(X, 18..21) and lives(X, Lausanne) -> profession(X, student)
which is more compact but requires a different discretization. There exist specialized techniques to deal with this problem (e.g. ARCS).
Components of a Data Mining Algorithm

- **Model or pattern structure**
  - Which kind of global model or local pattern is searched
  - Vector representation of documents
  - Association Rules

- **Score function**
  - Determine how well a given data set fits a model or pattern
  - Similarity of vectors
  - Support and confidence

- **Optimization and search method**
  - Finding best parameters for a global model: optimization problem
  - Finding data satisfying a pattern: search problem
  - Search the k nearest neighbors
  - Joining and pruning

- **Data management strategy**
  - Handling very large datasets
  - Inverted files
  - Sampling, partitioning and transaction elimination

We illustrate here of how the four main components of data mining algorithms, are instantiated with association rule mining. Compare also to the corresponding methods used for vector space retrieval.
Summary

• What is the meaning of support and confidence for an association rule?

• Is a high support for $A \cup B$ a sufficient condition for $A \rightarrow B$ or $B \rightarrow A$ being an association rule?

• Which properties on association rules and itemsets does the Apriori algorithm exploit?

• Which candidate itemsets can in the Apriori algorithm be eliminated in the pruning step and which during the database scan?

• How often is a database scanned when executing Apriori?

• How are association rules derived from frequent itemsets?
References

• Textbook
  - Jiawei Han, Data Mining: concepts and techniques, Morgan Kaufman, 2000,
    ISBN 1-55860-489-8

• Some relevant research literature
  - R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between
    sets of items in large databases. SIGMOD’93, 207-216, Washington, D.C.