Data Mining: Concepts and Techniques

Data Mining Primitives, Languages, and System Architectures — Chapter 4 —
Chapter 4: Data Mining Primitives, Languages, and System Architectures

- Data mining primitives: What defines a data mining task?
- A data mining query language
- Design graphical user interfaces based on a data mining query language
- Architecture of data mining systems
- Summary
Why Data Mining Primitives and Languages?

- Finding all the patterns autonomously in a database? — unrealistic because the patterns could be too many but uninteresting
- Data mining should be an interactive process
  - User directs what to be mined
- Users must be provided with a set of **primitives** to be used to communicate with the data mining system
- Incorporating these primitives in a **data mining query language**
  - More flexible user interaction
  - Foundation for design of graphical user interface
  - Standardization of data mining industry and practice
What Defines a Data Mining Task?

- Task-relevant data
- Type of knowledge to be mined
- Background knowledge
- Pattern interestingness measurements
- Visualization of discovered patterns
Task-Relevant Data (Minable View)

- Database or data warehouse name
- Database tables or data warehouse cubes
- Condition for data selection
- Relevant attributes or dimensions
- Data grouping criteria
Task-Relevant Data (Minable View)

- The virtual relations are called **views** in terms of database, the set of task-relevant data for data mining is called a **Minable view**.

- If a data mining task is to study associations between items frequently purchased at *AllElectronics* by customers in Canada, the task relevant data can be specified by providing the following information:
  - Name of the *database or data warehouse* to be used (e.g., *AllElectronics_db*)
  - Names of the *tables or data cubes* containing relevant data (e.g., *item, customer, purchases and items_sold*)
  - *Conditions* for selecting the relevant data (e.g., retrieve data pertaining to purchases made in Canada for the current year)
  - The *relevant attributes or dimensions* (e.g., *name* and *price* from the *item* table and income and age from the customer table)
Types of knowledge to be mined

- It is important to specify the knowledge to be mined, as this determines the data mining function to be performed.
- Kinds of knowledge include concept description, association, classification, prediction and clustering.
- User can also provide pattern templates. Also called metapatterns or metarules or metaqueries to guide the discovery process.
- A user studying the buying habits of AllElectronics customers may choose to mine association rules of the form:
  \[ P(X: \text{customer}, W) \land Q(X, Y) \Rightarrow \text{buys}(X, Z) \]
  Meta rules such as the following can be specified:
  \[ \text{age}(X, \text{“30…..39”}) \land \text{income}(X, \text{“40k….49K”}) \Rightarrow \text{buys}(X, \text{“VCR”}) \quad [2.2\%, 60\%] \]
  \[ \text{occupation}(X, \text{“student”}) \land \text{age}(X, \text{“20…..29”}) \Rightarrow \text{buys}(X, \text{“computer”})[1.4\%, 70\%] \]
Background Knowledge: Concept Hierarchies

- It is the information about the domain to be mined.
- Concept hierarchy
  - is a powerful form of background knowledge.
  - Defines a sequence of mappings from a set of low-level concepts to higher-level (more general) concepts.
  - Allows data to be mined at multiple levels of abstraction.
  - These allow users to view data from different perspectives, allowing further insight into the relationships.
Concept Hierarchy of the dimension Location

- **Level 0:**
  - All

- **Level 1:**
  - **Canada**
    - British Columbia
    - Ontario
  - **USA**
    - New York
    - Illinois

- **Level 2:**
  - New York
    - Buffalo
  - Illinois
    - Chicago

- **Level 3:**
  - Vancouver
  - Victoria
  - Toronto
  - Ottawa
Background Knowledge: Concept Hierarchies

- Rolling Up - Generalization of data
  - Allows to view data at more meaningful and explicit abstractions.
  - Makes it easier to understand
  - Compresses the data
  - Would require fewer input/output operations
- Drilling Down - Specialization of data
  - Concept values replaced by lower level concepts
- There may be more than one concept hierarchy for a given attribute or dimension based on different user viewpoints.
- Example:
  Regional sales manager may prefer the previous concept hierarchy but marketing manager might prefer to see location with respect to linguistic lines in order to facilitate the distribution of commercial ads.
Background Knowledge: Concept Hierarchies

- **Schema hierarchy**
  - E.g., street < city < province_or_state < country

- **Set-grouping hierarchy**
  - E.g., {20-39} = young, {40-59} = middle_aged

- **Operation-derived hierarchy**
  - email address: dmbook@cs.sfu.ca
    login-name < department < university < country

- **Rule-based hierarchy**
  - low_profit_margin (X) <= price(X, P_1) and cost (X, P_2)
    and (P_1 - P_2) < $50
Schema Hierarchies

- Schema hierarchy is the total or partial order among attributes in the database schema.

- May formally express existing semantic relationships between attributes.

- Provides metadata information.

- Example: location hierarchy
  
  \[ \text{street} < \text{city} < \text{province/state} < \text{country} \]
Set-grouping Hierarchies

- Organizes values for a given attribute into groups or sets or range of values.
- Total or partial order can be defined among groups.
- Used to refine or enrich schema-defined hierarchies.
- Typically used for small sets of object relationships.
- Example: Set-grouping hierarchy for age

{young, middle_aged, senior} all (age)
{20….29} young
{40….59} middle_aged
{60….89} senior
Operation-derived Hierarchies

- Operation-derived hierarchy is based on operations specified by users or data mining system.
- The operations may include:
  - decoding of information-encoded strings
  - information extraction from complex data objects
  - data clustering

Example: URL or email address
xyz@cs.iitm.in gives login name < dept. < univ. < country
Rule-based Hierarchies

- Rule-based:
  Occurs when either whole or portion of a concept hierarchy is defined as a set of rules and is evaluated dynamically based on current database data and rule definition.

Example: Following rules are used to categorize items as low_profit, medium_profit and high_profit_margin.

\[
\begin{align*}
\text{low_profit_margin}(X) &\leq \text{price}(X,P1)\cdot\text{cost}(X,P2)\cdot((P1-P2)<50) \\
\text{medium_profit_margin}(X) &\leq \text{price}(X,P1)\cdot\text{cost}(X,P2)\cdot((P1-P2)\geq50)\cdot((P1-P2)\leq250) \\
\text{high_profit_margin}(X) &\leq \text{price}(X,P1)\cdot\text{cost}(X,P2)\cdot((P1-P2)>250)
\end{align*}
\]
Measurements of Pattern Interestingness

- Simplicity
  e.g., (association) rule length, (decision) tree size

- Certainty
  e.g., confidence, \( P(A \mid B) = \frac{\#(A \text{ and } B)}{\#(B)} \), classification reliability or accuracy, certainty factor, rule strength, rule quality, discriminating weight, etc.

- Utility
  potential usefulness, e.g., support (association), noise threshold (description)

- Novelty
  not previously known, surprising (used to remove redundant rules, e.g., Canada vs. Vancouver rule implication support ratio)
Visualization of Discovered Patterns

- Different backgrounds/usages may require different forms of representation
  - E.g., rules, tables, crosstabs, pie/bar chart etc.
- Concept hierarchy is also important
  - Discovered knowledge might be more understandable when represented at high level of abstraction
  - Interactive drill up/down, pivoting, slicing and dicing provide different perspectives to data
- Different kinds of knowledge require different representation: association, classification, clustering, etc.
Chapter 4: Data Mining Primitives, Languages, and System Architectures

- Data mining primitives: What defines a data mining task?
- A data mining query language
- Design graphical user interfaces based on a data mining query language
- Architecture of data mining systems
- Summary
A Data Mining Query Language (DMQL)

- **Motivation**
  - A DMQL can provide the ability to support ad-hoc and interactive data mining
  - By providing a standardized language like SQL
    - Hope to achieve a similar effect like that SQL has on relational database
    - Foundation for system development and evolution
    - Facilitate information exchange, technology transfer, commercialization and wide acceptance

- **Design**
  - DMQL is designed with the primitives described earlier
Syntax for DMQL

- Syntax for specification of
  - task-relevant data
  - the kind of knowledge to be mined
  - concept hierarchy specification
  - interestingness measure
  - pattern presentation and visualization
- Putting it all together—a DMQL query
Syntax: Specification of Task-Relevant Data

- This involves specifying the database and tables or data warehouse containing the relevant data, conditions for selecting and relevant attributes or dimensions for exploration, and instructions regarding the ordering or grouping of the data retrieved.

- `use database database_name`, or `use data warehouse data_warehouse_name`

- `from relation(s)/cube(s) [where condition]`

- `in relevance to att_or_dim_list`

- `order by order_list`

- `group by grouping_list`

- `having condition`
Example: Specification of task-relevant data

Example 4.11 This example shows how to use DMQL to specify the task-relevant data described in Example 4.1 for the mining of associations between items frequently purchased at AllElectronics by Canadian customers, with respect to customer income and age. In addition, the user specifies that she would like the data to be grouped by date. The data are retrieved from a relational database.

```
use database AllElectronics.db
in relevance to I.name, I.price, C.income, C.age
from customer C, item I, purchases P, items_sold S
where I.item_ID = S.item_ID and S.trans_ID = P.trans_ID and P.cust_ID = C.cust_ID
      and C.address = "Canada"
group by P.date
```
Syntax: Kind of knowledge to Be Mined

- **Characterization**

The Mine_Knowledge_Specification indicates the data mining functionality to be performed.

Mine_Knowledge_Specification ::=  

*mine characteristics [as pattern_name]*  
*analyze measure(s)*

- Specifies the characteristic descriptions are to be mined.
- analyze clause is used for characterization, specifies aggregate measures, such as *count, sum* or *count%*

E.g.

```
mine characteristics as customerPurchasing analyze count%
```
Syntax: Kind of knowledge to Be Mined

- **Discrimination**
- Specifies the discriminant descriptions to be mined. These compare a given target class of objects with one or more other contrasting classes.

\[
\text{Mine\_Knowledge\_Specification ::= mine comparison [as pattern\_name] for target\_class where target\_condition \{versus contrast\_class\_i where contrast\_condition\_i\}}\text{ analyze measure(s)}
\]

E.g. mine comparison as purchaseGroups for bigSpenders where avg(l.price) >= $100 versus budgetSpenders where avg(l.price) < $100 analyze count
Syntax: Kind of Knowledge to Be Mined

- **Association**
  - Specifies the mining of patterns of association.
    - Mine_Knowledge_Specification ::= mine associations [as pattern_name]
      [matching <metapattern>]
  - The user can specify metapatterns or metarules with the `matching` clause.
    - E.g. mine associations as buyingHabits
      matching P(X:custom, W) ^ Q(X, Y)=>buys(X, Z)
Syntax: Kind of Knowledge to Be Mined

- **Classification**
  - Specifies the patterns for data classification are to be mined.
  - The `analyze` clause specifies that the classification is performed according to the values of `classifying_attribute_or_dimension`.
  - For categorical attributes or dimensions, each value represents a class such as "low_risk", "medium_risk" and "high_risk", for the attribute `credit_rating`.
  - For numeric attributes or dimensions, each class may be defined by a range of values such as "20-39", "40-59", "60-89" for age.

  **E.g.** `mine classification as classifyCustomerCreditRating analyze credit_rating`

- **Other Patterns**
  - clustering, outlier analysis, prediction ...
Syntax: Concept Hierarchy Specification

- Concept hierarchies allow the mining of knowledge at multiple levels of abstraction.
- To specify what concept hierarchies to use
  
  \[
  \text{use hierarchy } \textit{<hierarchy>} \text{ for } \textit{<attribute_or_dimension>}
  \]
- We use different syntax to define different type of hierarchies
  - schema hierarchies
    
    define hierarchy \textit{time\_hierarchy} on \textit{date} as [date,month quarter,year]
  - set-grouping hierarchies
    
    define hierarchy \textit{age\_hierarchy} for \textit{age} on \textit{customer} as
    
    level1: \{young, middle\_aged, senior\} < level0: all
    level2: \{20, ..., 39\} < level1: young
    level2: \{40, ..., 59\} < level1: middle\_aged
    level2: \{60, ..., 89\} < level1: senior
Concept Hierarchy Specification (Cont.)

- operation-derived hierarchies
  
  define hierarchy **age_hierarchy** for **age** on **customer** as
  
  \{age\_category(1), \ldots, age\_category(5)} :=
  
  cluster(default, age, 5) < all(age)

- rule-based hierarchies
  
  define hierarchy **profit\_margin\_hierarchy** on **item** as
  
  level\_1: low\_profit\_margin < level\_0: all
  
  if (price - cost) < $50

  level\_1: medium-profit\_margin < level\_0: all
  
  if ((price - cost) > $50) and ((price - cost) <= $250)

  level\_1: high\_profit\_margin < level\_0: all
  
  if (price - cost) > $250
The user can help control the number of uninteresting patterns returned by the data mining system by specifying measures of pattern interestingness and their corresponding thresholds.

Interestingness measures and thresholds can be specified by a user with the statement:

```
with <interest_measure_name> threshold = threshold_value
```

Example:

```
with support threshold = 0.05
with confidence threshold = 0.7
```
Specifying Pattern Presentation

This is to specify the display of discovered patterns in one or more forms, including rules, tables, crosstabs, pie or bar charts, decision trees, cubes, curves or surfaces.

Specify the display of discovered patterns

\[ \text{display as } <\text{result_form}> \]

To facilitate interactive viewing at different concept level, the following syntax is defined:

\[ \text{Multilevel_Manipulation ::= } \text{roll up on attribute_or_dimension} \]
\[ \text{drill down on attribute_or_dimension} \]
\[ \text{add attribute_or_dimension} \]
\[ \text{drop attribute_or_dimension} \]
Putting it all together: A DMQL query

Suppose, as a marketing manager of AllElectronics, you would like to characterize the buying habits of customers who purchase items priced at no less than $100, with respect to the customer's age, the type of item purchased, and the place in which the item was made. For each characteristic discovered, you would like to know the percentage of customers having that characteristic. In particular, you are only interested in purchases made in Canada, and paid for with an American Express ("AmEx") credit card. You would like to view the resulting descriptions in the form of a table. This data mining query is expressed in DMQL as follows.

use database AllElectronics_db
use hierarchy location_hierarchy for B.address
mine characteristics as customerPurchasing
analyze count%
in relevance to C.age, I.type, I.place_made
from customer C, item I, purchases P, items_sold S, works_at W, branch
where I.item_ID = S.item_ID and S.trans_ID = P.trans_ID
    and P.cust_ID = C.cust_ID and P.method_paid = "AmEx"
    and P.empl_ID = W.empl_ID and W.branch_ID = B.branch_ID and
    B.address = "Canada" and I.price >= 100
with noise threshold = 0.05
display as table
Other Data Mining Languages & Standardization Efforts

- Association rule language specifications
  - MSQL (Imielinski & Virmani’99)
  - MineRule (Meo Psaila and Ceri’96)
  - Query flocks based on Datalog syntax (Tsur et al’98)
- OLEDB for DM (Microsoft’2000)
  - Based on OLE, OLE DB, OLE DB for OLAP
  - Integrating DBMS, data warehouse and data mining
- CRISP-DM (CRoss-Industry Standard Process for Data Mining)
  - Providing a platform and process structure for effective data mining
  - Emphasizing on deploying data mining technology to solve business problems
Chapter 4: Data Mining Primitives, Languages, and System Architectures

- Data mining primitives: What defines a data mining task?
- A data mining query language
- Design graphical user interfaces based on a data mining query language
- Architecture of data mining systems
- Summary
Designing Graphical User Interfaces Based on a Data Mining Query Language

- What tasks should be considered in the design GUIs based on a data mining query language?
  - Data collection and data mining query composition
  - Presentation of discovered patterns
  - Hierarchy specification and manipulation
  - Manipulation of data mining primitives
  - Interactive multilevel mining
  - Other miscellaneous information
Chapter 4: Data Mining Primitives, Languages, and System Architectures

- Data mining primitives: What defines a data mining task?
- A data mining query language
- Design graphical user interfaces based on a data mining query language
- Architecture of data mining systems
- Summary
Data Mining System Architectures

- Information processing and data analysis infrastructures have been developed on database systems and data warehouses.
- These include accessing, integration, consolidation, and transformation of multiple heterogeneous databases, ODBC/OLEDB connections, Web-accessing and reporting and OLAP analysis tools.
- Coupling schemes for data mining system with DB/DW system
  - No coupling—flat file processing, not recommended
  - Loose coupling
  - Semi-tight coupling—enhanced DM performance
  - Tight coupling—A uniform information processing environment
No Coupling

- The Data Mining system will not utilize any function of a Database or Data Warehousing system.
- It fetch data from file system process data using data mining algorithms and store the results in another file.

**Drawbacks**

- Without using DB/DW system, a DM system may spend a substantial amount of time in finding, collecting, cleaning and transforming data.
- There are many tested, scalable algorithms and data structures implemented in DB and DW systems. Without any coupling of such systems, a DM system will need to use other tools to extract data.
Loose Coupling

- In this DM system will use some facilities of DB/DW system, fetching data from a data repository managed by these systems, performing data mining, and then storing the mining results either in a file or in a designated place in a database or data warehouse.
- By using query processing, indexing and other system facilities loose coupling can fetch any portion of data stored in DB/DW.
- Many loose coupled mining system are main memory-based.
- It is difficult for loose coupling to achieve high scalability and good performance with large data sets.
Semi-tight Coupling

- In this besides linking of DM to a DB/DW system, efficient implementations of a few essential data mining primitives – sorting, indexing, aggregation, histogram analysis, multiway join and precomputation of some essential statistical measures, such as sum, count, max, min and so on can be provided in the DB/DW system.
- Frequently used intermediate mining results can be precomputed and stored in the DB/DM system.
- This design can enhance the performance of a DM system because of precomputed results.
Tight Coupling

- Tight coupling means that the DM system is smoothly integrated into the DB/DW system.
- Data mining queries and functions are optimized based on mining query analysis, data structures, indexing schemes, and query processing methods of a DB or DW system.
- With further technology advances, DM, DB and DW systems will evolve and integrate together as one information system with multiple functionalities.
- This approach is highly desirable since it facilitates efficient implementation of data mining functions, high system performance, and an integrated information processing environment.
Chapter 4: Data Mining Primitives, Languages, and System Architectures

- Data mining primitives: What defines a data mining task?
- A data mining query language
- Design graphical user interfaces based on a data mining query language
- Architecture of data mining systems
- Summary
Summary

- Five primitives for specification of a data mining task
  - task-relevant data
  - kind of knowledge to be mined
  - background knowledge
  - interestingness measures
  - knowledge presentation and visualization techniques to be used for displaying the discovered patterns

- Data mining query languages
  - DMQL, MS/OLEDB for DM, etc.

- Data mining system architecture
  - No coupling, loose coupling, semi-tight coupling, tight coupling
References

Thank you !!!