The MiningMart Approach

Katharina Morik Martin Scholz
Department of Computer Science, Universität Dortmund
{morik,scholz}@ls8.cs.uni-dortmund.de

1 Goal and Objectives of the MiningMart Project

The use of very large databases has enhanced in the last years from supporting transactions to additionally reporting business trends. The interest in analysing the data has increased. One important topic is customer relationship management with the particular tasks of customer segmentation, customer profitability, customer retention, and customer acquisition (e.g. by direct mailing). Other tasks are the prediction of sales in order to minimize stocks, the prediction of electricity consumption or telecommunication services at particular day times in order to minimize the use of external services or optimize network routing, respectively. The health sector demands several analysis tasks for resource management, quality control, and decision making.

On-line Analytical Processing (OLAP) offers interactive data analysis by aggregating data and counting the frequencies. This already answers questions like the following:
What are the attributes of my most frequent customers?
Which are the frequently sold products?
How many unpaid bills do I have to expect per year?
How many returns did I receive after my last direct mailing action?

Reports that support decision making need more detailed information. Questions are more specific, for instance:
Which customers are most likely to sell their insurance contract back to the insurance company before it ends?
How many sales of a certain item do I have to expect in order to not offer empty shelfs to customers and at the same time minimize my stock?

Knowledge Discovery in Databases (KDD) can be considered a high-level query language for relational databases that aims at generating sensible reports such that a company may enhance its performance. The high-level question is answered by a *data mining* step. Several data mining algorithms exist. However, before they can be applied, several steps need to be done: data cleaning and handling of null values, feature generation and selection (in databases this means to construct additional columns and select the relevant attributes), selection of an appropriate data mining algorithm for the task, transformation of the data into the input format of the algorithm. If we inspect real-world applications of knowledge discovery, we realize that up to 80 percent of the efforts are spent on the pairing of data with

algorithms and the clever preprocessing of the data. Moreover, most tools for data analysis need to handle the data themselves and cannot access a database directly. Sampling the data and converting them into the desired format enhances the efforts further. Hence, KDD can only become an actual high-level query language accessing real world databases or data warehouses, if preprocessing and the selection of a data mining algorithm is eased and enhanced. This is the main goal of the MiningMart project [1].

What is MiningMart's path to reaching the goal? A first step is to implement operators that perform data transformations such as, e.g., discretization, handling null values, aggregation of attributes into a new one, or collecting sequences from time-stamped data. The operators directly access the database and are capable of handling large masses of data. Machine learning is not restricted to the data mining step, but is also applicable in preprocessing. This view offers a variety of learning tasks that are not as well investigated as is learning classifiers. For instance, an important task is to acquire events and their duration (i.e. a time interval) on the basis of time series (i.e. measurements at time points).

Given database oriented operators for preprocessing, the second step is to develop and collect successful cases of knowledge discovery. Since most of the time is used to find chains of operator applications that lead to good answers to complex questions, it is cumbersome to develop such chains over and over again for very similar discovery tasks and data. Currently, even the same task on data of the same format is implemented anew every time new data are to be analysed. Therefore, the re-use of successful cases would speed up the process considerably. The particular approach of the MininjgMart project is to allow the re-use of cases by means of meta-data, also called *ontologies*. Meta-data describe the data as well as the operator chains. A compiler generates the SQL code according to the meta-data. The advantages of meta-data driven software generation are:

Data documentation: The data together with their statistics and important features for data mining (e.g., presence of null values) are well documented. This extends the meta-data as are usual in relational databases.

Case documentation: The chain of preprocessing operators including all parameter settings is documented, as well. This makes a case reproducable. In contrast, the current state of documentation is most often the memory of the particular scientist who developed the case.

Abstraction: Meta-data are given at different levels of abstraction, a conceptual (abstract) and a relational (executable) level. The same case at the conceptual level can be mapped on several different cases at the relational level. This makes an abstract case re-usable.

Ease of case adaptation: In order to run a given sequence of operators on a new database, only the relational meta-data and their mapping on the conceptual meta-data has to be written.

The MiningMart project has developed a model for meta-data together with its compiler and implements human-computer interfaces that allow database managers and case design-

[1] The MiningMart project is supported by the European Union under the contract IST-11993.

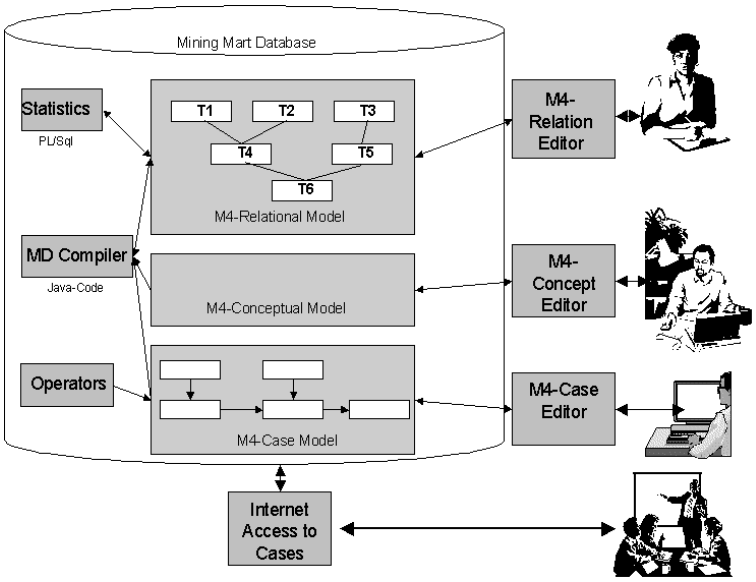

Figure 1: Overview of the MiningMart system

ers to fill in their application-specific meta-data. The system will support preprocessing and can be used stand-alone or in combination with a toolbox for the data mining step.

2 System Architecture

This section gives an overview of how a case is represented at the meta-level, how it is practically applied to a database, and which steps need to be performed.

2.1 The Meta-Model of Meta-Data M4

The form in which meta-data are to be written is specified in the meta-model of meta-data, M^4. It is structured along two dimensions, topic and abstraction. The *topic* is either the business data or the case. The business data are the ones to be analysed. The case is a sequence of (preprocessing) steps. The *abstraction* is either conceptual or relational. Where the conceptual level is expected to be the same for various applications, the relational level actually refers to the particular database at hand. The meta-data written in the form as specified by M^4 are stored in a relational database themselves.

2.2 Editing the Conceptual Data Model

As depicted in figure 1, there are different kinds of experts working at different ends of a knowledge discovery process. First of all a domain expert will define a conceptual data model, using a concept editor. The entities involved in data mining are made explicit by this expert. The conceptual model of M^4 is about *concepts* having *features* and *relationships* between these concepts. Concepts and Relationships may be organized hierarchically by means of inheritance. Examples for concepts are "Customer" and "Product", a relationship between these two could be "Buys".

2.3 Editing the Relational Model

Given a conceptual data model, a database administrator maps the involved entities to the corresponding database objects. The relational data model of M^4 is capable of representing all the relevant properties of a relational database. The most simple mapping from the conceptual to the relational level is given, if concepts directly correspond to database tables or views. This can always be achieved manually by inspecting the database and creating a view for each concept. However, more sophisticated ways of graphically selecting attributes (or "features") and aggregating them to concepts, increase the acceptance by end users. In the project, the relational editor is intended to support this kind of activity. In general it should be possible to map all reasonable representations of entities to reasonable conceptual definitions. A simple mapping of the concept "Customer", containing the features "Customer ID", "Name", "Address" to the database would be to state that the table "CUSTOMER" holds all the necessary attributes, e.g. "CUSTOM_ID", "CUST_NAME" and "CUST_ADDR". An example for more complex mappings occurs if the information about name and address needs to be joined first, e.g. using the shared key attribute "CUSTOM_ID".

2.4 The Case and Its Compiler

All the information about the conceptual descriptions and about the according database objects involved are represented within the M^4 model and stored within relational tables. "M^4-Cases" denote a collection of steps, basically performed sequentially, each of which changes or augments one or more concepts. Each step is related to exactly one *operator*, and holds all of its input arguments. The M^4 compiler reads the specifications of steps and starts the according executable operator, passing all the necessary inputs to it. Depending on the operator, the database might be altered. In any case the M^4 meta-data will have to be updated by the compiler. A machine learning tool to replace missing values is an example for operators altering the database. In contrast, for operators like a join it is sufficient to *virtually* add the resulting view – together with corresponding SQL-statement – to the meta-data.

The task of a case designer, ideally a data mining expert, is to find sequences of steps resulting in a representation well suited for data mining. This work is supported by a special tool, the case editor. Usually a lot of different operators will be involved in a case of preprocessing steps. A list of available operators and their overall categories, e.g. Feature Construction, Clustering or Sampling is part of the conceptual case model M^4. In every step the case designer chooses an applicable operator, sets all the parameters of the operator, assigns the input concepts, input attributes and/or input relations and gives some specifics about the output. Applicability conditions are considered in two ways. On one hand, constraints of operators can be checked on the basis of meta-data. These are, for instance, the presence or absence of NULL values. On the other hand, the conditions of operators can be checked on the basis of the business data when running the case. Applicability constraints and conditions support the case designer by checking the validity of a sequence of steps while it is created.

The sequence of many steps, namely a case, transforms the original database into another representation. Each step and their ordering is formalized within M^4, so the system is automatically keeping track of the performed activities. This enables the user to interactively edit and replay the case or parts of it. Further more, as soon as an efficient chain of preprocessing has been found, it can easily be exported by just submitting the conceptual meta-data.

3 The Case Base

One of the project's objectives is to set up a case-base of successful cases on the internet. The shared knowledge allows all internet users to benefit from a new case. Submitting a new case of best practice is a safe advertisement for KDD specialists or service providers, since the relational data model is kept private. Only the conceptual and the case model is published. Currently, the project as developed three cases, analsis for direct mailing, for fraud detection in telecommunication, and for sales prediction.

To support users in finding the most relevant cases, their inherent structure will be exploited. An according internet interface will be set up, visualizing the conceptual meta-data. It will be possible to navigate through the case-base and investigate single steps i.e., which operators were used on which kind of concepts. The internet interface is supposed to read the data directly from the M^4 tables in the database, avoiding redundancies.

Additionally to the data explicitly represented in M^4, a business level is added. This level aims at relating the case to business goals and to give several kinds of additional descriptions, like which success criteria were important for the case. Especially the more informal descriptions should help decision makers to find a case tailored for their specific domain and problem. The additional information is stored in an XML-representation, directly connected[2] to the M^4 entities.

It will be possible to start the search for a case at each category of the business level

[2] In the visual form on the web these connections are hyperlinks.

or conceptual level. In this sense the cases are indexed by all the categories part of the conceptual M^4 model and the business model. If a user considers a case useful, then its conceptual data can be downloaded from the server. The downloadable case itself will be a category in the XML framework. Up to now SQL-Scripts installing the meta-data into the user's M^4 tables were found to be the most practical solution to support case downloads. If problems arise, or further help is necessary, the business level holds a category for the case designer or the company providing service.

4 Conclusion

The relevance of supporting not only single steps of data anlysis but sequences of steps has long been underestimated. Where a large variety of excellent tools exist which offer algorithms for a data mining step, only very few approaches exist which tackle the task of making clever choices during preprocessing and combining these choices to an effective and efficient sequence. The CLEMENTINE system offered processing chains to users. However, the focus lies on the data mining step, not the preprocessing chain. The common data format in tool boxes such as, e.g. SPSS or WEKA provides users with the prerequisites to formulate their own sequences [WF00]. However, the user is programming the sequence and has to do this anew for very similar tasks. The recent IDEA system is most similar to the MiningMart approach [BHP02]. Chains of operators are composed accoring to a ranking of algorithms in order to detect the best choice of an algorithm given data characteristics. Meta-data describing the data as well as the algorithms are used in order to check the validity of operator sequences or incorporate an additional step which allows to apply the best operator. The difference lies first in MiningMart's orientation towards very large databases. IDEA uses the WEKA data format and, hence, is restricted to smaller files. The data transformations and aggregations incorporated as manual operators in the MiningMart system are not taken into account in IDEA, because they are not needed in the single table small sample representation of WEKA tools. The second distinction is the approach to determining the best sequence of preprocessing. Although the MiningMart system exploits applicability conditions of operators in order to check the validity of sequences, it does not aim at planning the best sequence or perform a ranking of possible algorithms at each step of an operator chain, as IDEA can do. Instead, MiningMart exploits the findings of expert case designers. Real-world application of knowledge discovery comprise hundreds of steps in a KDD run (including manual operators) and ranking every algorithm at each of the steps would exhaust computing capacity. We feel that the adaptation of excellently solved KDD problems best combines human expertise and computational power.

BHP02 Abraham Bernstein, Shawndra Hill, and Foster Provost. An Intelligent Assitsant for the Knowledge Discovery Process. Technical report, New York University, Leonard Stern School of Business, 2002.

WF00 Ian Witten and Eibe Frank. *Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.

Workshop

Sicherheitsrelevante Software in industriellen Anwendungen

Leitung und Durchführung
Wolfgang D. Ehrenberger, Gerhard H. Schildt