A Framework for Data Mining and KDD[*]

Ingolf Geist
University of Magdeburg
P.O. Box 4120
D-39016 Magdeburg, Germany
geist@iti.cs.uni-magdeburg.de

ABSTRACT

The KDD process is a non-trivial, iterative, interactive and multi-step process, that requires the development of a unifying model. This model have to ensure an uniform description of data and patterns and the control of the manipulation of the data and patterns. Thus, the model defines operations within the pattern and data, as well as transition operations between data and patterns.

This paper proposes a framework consisting of a model view, a data view and a process view. It focuses on the model and data view. The model view contains a set of *mining models*, which contain all information of a data mining result, that are based on constraints. The proposed model algebra uses concepts of constraint databases as well as collective and parallel data mining. The whole process is supported by using operations between data and model view.

Keywords

Knowledge Discovery in Databases, Constraint Databases, Mining Model

1. INTRODUCTION

Knowledge Discovery in Databases (KDD) is a non-trivial, multi-step process [6]. The process consists of data integration, preparation and transformation, data mining as well as evaluation and presentation of the data mining results. These steps are processed iteratively and interactively. An analyst uses different techniques and approaches during the data analysis. For instance, the user can at first create a classification model for customers, afterwards he uses the created model to get all "highrisk" customers. The next step could be the analysis of the buying behavior of this customer group by means of a association rule analysis.

A KDD process requires an efficient exchange of mining models and data sets between different analysis methods and programs. Therefore, there is a need for a unifying framework, that provides a uniform data structure for all data mining patterns, operations to manipulate these patterns as well as the transition to the data level and from the data level. Furthermore, the data level should be included in the framework, because the preparation and integration of data takes a lot of time during the data analysis. The third part is a control mechanism managing the whole process by using the operation of the other parts.

[*]Research was supported by the grant FOR 345/1 from the German Research Council(DFG).

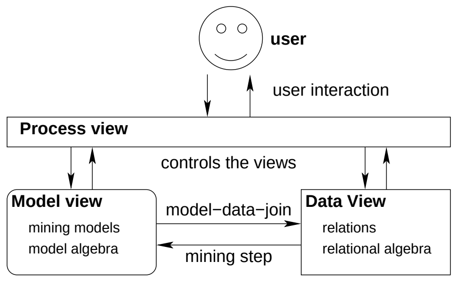

Figure 1: Three view architecture

A three view architecture is proposed, that consists of *process view*, *model view* and *data view* and operations between the views, to meet these requirements. Figure 1 illustrates the architecture.

The user interacts with the *process view*, which offers operations supporting management and controlling of KDD processes. It combines different processes and results by using the methods offered by the *data view* and the *model view*. Different presentation modes support the analysis tasks. Iteration, condition, join and split nodes are used as control mechanisms. The control mechanisms of the process view are not discussed in this paper, but rather the foundations of the model view and the data view.

The raw data sets are described in the *data view*. The relational data model is used in the proposed architecture, because it provides a formal foundation for the data view and provides the relational algebra to manipulate the data objects.

The third view in our framework is the *model view*. The model view consists of a set of *mining models* which contains all information about a mining result. A mining model contains the patterns, information about the pattern, e.g. an interestingness function as well as additional properties. Properties are for instance class labels or cluster information like means or medians. The patterns are modeled as tuples of linear inequality constraints. This kind of modeling allows a unified presentation of the results of mining algorithms. Based on this model structure an algebra is defined which provides operations for modification and combination of mining models.

1.1 Outline of the paper

Following the introduction in Section 1 the remaining part of the paper focuses on the model view. In the following Section 2 the use of constraints as pattern language is motivated and basic concepts of constraint databases are introduced. In Section 3 at first the proposed data model of the model view is informally discussed and a formalization of the mining models is shown afterwards. The algebra within the model view data model is introduced in section 4. Section 5 provides an overview of the related work. The paper ends with a conclusion and an outlook to the future work and possible optimization issues in the framework.

2. CONSTRAINT DATABASES AND THEIR APPLICATION IN KDD

Before the discussion of the framework an introduction is given to concepts of constraint databases and their application in the KDD area. In data mining the mined patterns are very different, but every pattern describes a possible subset of the data. Constraints are used to describe possible infinite data sets, thus, they are a natural choice for description of results of data mining algorithms. The concepts of constraint databases provide a powerful basis for manipulation these patterns.

2.1 Constraints and patterns

First we define the concepts *constraint* and *constraint k-tuple* as they are used in the constraint database research area. A constraint can be used to describe a possible infinite data set. Linear inequality constraints, a special class of constraints, are sufficient to describe mining results, as they can represent most of the used shapes [11]. Commonly, they are defined as:

Definition 1. Let Ω be a vocabulary. A *constraint* over Ω is an atomic first-order formula over Ω, or the negation of an atomic formula [19]. Assuming the attribute set $R = \{A_1, \ldots, A_k\}$ [19]. A *linear inequality constraint* over R is a condition $p \leq c$, where p is a function $a_1A_1 + \ldots + a_kA_k + a_0$ with real coefficients.

Assume a relation with the schema $R = \{A_1, \ldots, A_k\}$. A constraint k-tuple is used to describe a subset of this relation. As the patterns can be convex or concave the notion of the *extended constraint k-tuple* [3] is used. Formally, the extended constraint k-tuple is defined as following.

Definition 2. A *constraint k-tuple*, in variables $A_1, \ldots, A_k$, over vocabulary Ω, is a finite conjunction $\phi_1 \wedge \ldots \wedge \phi_n$, where each ϕ_i, $for 1 \leq i \leq n$, is an Ω-constraint. Furthermore, the variables in each ϕ_i are among $A_1, \ldots, A_k$. An *extended constraint k-tuple* is defined as the constraint k-tuple except, $\wedge$ as well as $\vee$ are used as connectives between the constraints.

This definition allows the union of several constraint k-tuples in one extended constraint k-tuple, which can represent complex objects. In our case the extended constraint k-tuple consists of a disjunction of conjunctions of linear inequality constraints. Such a tuple describes one pattern of a mining result.

After introducing the extended k-tuples as pattern description their manipulation has to be discussed. For this purpose the concepts of *constraint databases* can be used, that are described in the next section.

2.2 Constraint Databases

Constraint databases(CDB) were introduced in [12]. One concept, the constraint k-tuple[1], is already defined in the previous section. As some concepts of CDB are used in the next sections, their definition is given here. Other important terms are defined for instance in [14].

Definition 3. A *constraint relation of the arity k*, over Ω, is the finite set $r = \{t_1, \ldots, t_m\}$, where each t_i, for $1 \leq i \leq m$, is a extended constraint k-tuple in the same variables $A_1, \ldots, A_k$. An *extension* for an extended k-tuple $t \in r$, denoted by $ext(t)$, is set of relational tuples, which makes the formula t true in a certain domain [19]. The *extension of a constraint relation* r is a nested relation $n = \{ext(t_1), \ldots, ext(t_m)\}$ [3].

As every tuple represents a pattern, the *nested relational* interpretation of the constraint relation is useful. A constraint relation is then the representation of a set of patterns. Within the nested interpretation an extended constraint algebra [3] is used to manipulate the constraint relation. In this algebra are besides tuple operations also set operations defined, which are used in the proposed framework, thus, the following definition gives an overview over selected operations.

Definition 4. Selected set and tuple operations of the extended constraint algebra [3]:
Set Selection: $\bar{\sigma}^s_{Q_1 \subseteq Q_2}(r_1)$ with $\alpha(Q_1) \subseteq \alpha(Q_2)$[2]:

$$r = \{t : t \in r_1, ext(Q_1(t)) \subseteq ext(\bar{\pi}_{[\alpha(Q_1)]}(Q_2(t)))\}$$

and $\sigma^s_{Q_1 \cap Q_2}(r_1)$ with $\alpha(Q_1) = \alpha(Q_2)$:

$$r = \{t : t \in r_1, ext(Q_1(t)) \cap ext(Q_2(t)) \neq \emptyset\}.$$

Set Difference: $r_1 \bar{\setminus}^s r_2$ with $\alpha(r) = \alpha(r_1) = \alpha(r_2)$:

$$r = \{t : t \in r_1, \nexists t' \in r_2 : ext(t) = ext(t')\}.$$

Projection: $\bar{\pi}_{[x_{i_1}, \ldots, x_{i_p}]}(r_1)$ with $\alpha(r_1) = \{x_1, \ldots, x_k\}$:

$$r = \{\pi_{[x_{i_1}, \ldots, x_{i_p}]}(t) : t \in r_1, ext(\pi_{[x_{i_1}, \ldots, x_{i_p}]}(t) \neq \emptyset)\}$$

In this section the foundations of constraint databases were introduced as used in the proposed model. In order to support data mining the foundations have to be extended to achieve a sufficient data model, which is described in the remaining part of the paper.

3. THE MODEL VIEW

The model view consists of a set of *mining models*. A mining model is supposed to contain all information about the results of a run of a data mining algorithm. It contains the patterns that are modeled as constraint k-tuples over

[1]In the remaining paper constraint k-tuple is equivalent to extended constraint k-tuple
[2]Q denotes either a tuple or the projection of tuple

the input attribute set. These constraint k-tuples describe the sets of the data points in the single patterns. An interestingness function value and properties are assigned to each pattern.

Assuming $R = \{A_1, \ldots, A_k\}$ is a set of attributes which describes the input data of the model, then a mining model schema is defined as following:

Definition 5. A *mining model schema* is an attribute set $M = \{C_R, I, L_1, \ldots, L_n\}$. C_R denotes the pattern attribute on R and its domain $dom(C_R)$ contains all possible constraint k-tuples in attribute set R. The interestingness function I maps a constraint k-tuple to the domain $[0, 1]$ according to the mined relation r over R. $L_1, \ldots, L_n$ are additional properties of the patterns, which can contain derived information.

Based on the definition of a model schema a model is defined as following: A model is an instance of a model schema and is created from the relation r with name R and the pattern description is a subset of all possible constraint k-tuples in C_R. Each pattern has a quality value w.r.t. r and additional properties.

Definition 6. A *mining model* instance over a relation r with schema R is defined as

$$\begin{aligned} m \quad = \quad & \{p \; : \; p[C_R] \in dom(C_R), \\ & p[I] = I(p[C_R], r) l_1 \in dom(L_1), \ldots, l_n \in dom(L_n)\}. \end{aligned}$$

The model view consists of a set of mining models, which is derived from the data sets of the data view. The schema are model names together with their attributes.

Definition 7. The schema of the *model view* is the set of model names and their assigned attributes. The *instance* of the model view is the set of models over the schema as defined above.

Assuming model m is a classification model. A classification pattern within the model is represented as the following:

$$p = (A_1 \leq 45 \wedge A_2 \geq 500, 0.98, \text{highrisk}).$$

A tuple t of a relation r belongs to class "highrisk", if $t[A_1] \leq 45 \wedge t[A_2] \geq 500$ evaluates to *true*. The accuracy in the training set is 98%.

A cluster description benefits from the extended k-tuple notion, as it allows the combination of several convex shapes by using a disjunction of conjunctions of constraints. For instance:

$$(c_{1_1} \wedge \ldots \wedge c_{n_1} \vee c_{1_2} \ldots \wedge c_{n_n}, 0.89, 34).$$

c denotes a constraint and 0.89 shows the cluster quality.

4. OPERATIONS

In the proposed framework the process view controls the other views, which provide operations on their elements and transition operations. Thus, there are operations on the mining models, the data as well as operations between data and models. The operations on the data are defined by the relational algebra and its extensions like grouping and aggregation.

The operations over models (Section 4.1) are based on the constraint algebra and the ideas of collective data mining [13] and distributed data mining[17]. The operations form an algebra and support the properties: *closure*, *efficiency* and a *minimum* of operations as well as *transition* between model and data view.

The operations *Selection* σ, *Projection* π, *Natural Join* $\bowtie$, *Union* $\cup$, *Difference* $-$ and *Renaming* β are used in the model algebra. In Section 4.2 the operations between the data view and the model view are defined. These transition operations are *mining* μ, that creates a model from the data and *model-data-join* $\bowtie_{md}$, which produces the extensions for a model.

4.1 Operations on Mining Models

4.1.1 Selection

The selection operation enables the analyst to select the most interesting patterns in his view. There are three possibilities of selections of a model: constraint selection, interestingness selection and label selection. The constraint selection extracts tuples from the model using the set selection. Using the set selection it is possible to define the spatial relationships *intersection*, *disjunction*, *containment* and *equivalence* [3]. The second type of selection is the interestingness selection which selects the patterns according to their value of the interestingness function. A predicate $pred(I) = i(c, s) \; \theta \; k$ where $k \in [0, 1]$ and $\theta \in \{\leq, <, =, \neq, >, \geq\}$ is used as selection condition. Furthermore, we can use the label attributes in a relational selection condition. *cscond* denotes a constraint selection condition and *cond* is relational selection condition. Formally, the selection operation is defined as following. The term $m[C_R]$ denotes the projection to the constraint attribute of the model m.

Definition 8. Assuming the model m over the relation r with schema $\alpha(m) = \{C_R, I, \mathbf{L}\}$ and $\mathbf{L} = \{L_1, \ldots, L_n\}$. The *model selection* σ is defined as

$$\begin{aligned} \sigma_{cscond}(m) &= \{t \; : \; t \in p, t[C_R] \in \bar{\sigma}^s_{cscond}(m[C_R])\}) \\ \sigma_{pred(I)}(m) &= \{t \; : \; t \in m, \; pred(t[I]) \text{ is true}\} \\ \sigma_{cond(\mathbf{L})}(m) &= \{t \; : \; t \in p, \; cond(t[\mathbf{L}]) \text{ is true}\}. \end{aligned}$$

Assume the model m is representing cluster patterns and all clusters which are contained in data set described by $P = (100 \leq A_1 \leq 1000 \wedge 200 \leq A_2 \leq 2000)$ shall be selected. The according query is: $\sigma_{t \subseteq P}(m)$.

4.1.2 Projection

The *projection* operation is the second proposed operation, which modifies the patterns in the way, that the patterns are only valid over the projected data set. The constraint projection requires the recomputation of the interestingness values and the properties. The operation is necessary to support the union operation. The projection is restricted, so that always at least the interesting function values and patterns are existent. Assume a attribute set $R = \{A_1, \ldots, A_m, A_{m+1}, \ldots, A_k\}$.

Definition 9. Given a model m with schema $\alpha(m) = M = \{C_R, I, L_1, \ldots, L_l, L_{l+1}, \ldots, L_k\}$. The *projection* π is defined as:

$$\begin{aligned} \pi_{A_1,\ldots,A_m}(m) &= \{t \; : \; t[C_{A_1,\ldots,A_m}] \in \bar{\pi}_{A_1,\ldots,A_m}(C_R), \\ & \qquad t[I] = I(t[C_{A_1,\ldots,A_m}, r)\} \\ \pi_{L_1,\ldots,L_l}(m) &= \{t[C_R, I, L_1, \ldots, L_l] \; : \; t \in m\}. \end{aligned}$$

4.1.3 Natural Join

After the discussion of the operations that modify or query one model, operations are defined which combine models. The combining operations are quite different to the former operations because they include special algorithms which define their semantic. The natural join operation is the first operation to be discussed. Assuming there are two relations r_1 and r_2, over both relations a model is defined. Each of them has the same interestingness function and the same properties. Furthermore, it is required that $R_1 \cap R_2 \neq \emptyset$. Then we can use the information from both models, a sample of the combined data set and a collective data mining (CDM) algorithm [13] to create the joined mining model. Thus, the join operation is used to extend the description of the pattern by new attributes. The extension of the new model is defined by the data mining algorithm which performs the computation of the new patterns, the interestingness function as well as the labels. Each of the joined models have to be of the same type. The join operation and the special algorithms require further research in combining mining results.

Definition 10. Let R_1 and R_2 be attribute sets of the relations r_1 and r_2, respectively. Further it holds $R = R_1 \cup R_2$. And assuming m_1 and m_2 are mining models over r_1 and r_2 and it holds $I_1 = I_2$ and $\mathbf{L_1} = \mathbf{L_2}$. Then, the *natural join* $\bowtie$ operation is defined as:

$$\begin{aligned} m_1 \bowtie m_2 &= \{t : t[C_R] \in dom(t[C_{R_1 \cup R_2}]), \text{ the other} \\ &\quad \text{attributes are created by a} \\ &\quad \text{CDM algorithm}\} \\ \alpha(m_1 \bowtie m_2) &= \{C_R, I_1, \mathbf{L_1}\}. \end{aligned}$$

By this definition the semantic of the operation is depending on the implementing CDM algorithm, but the structure of the inputs and outputs can be defined, so the analyst is enabled to follow the operation.

4.1.4 Union

The join operation defines the extension of the attribute set of the model by combining two models over two relations. The *union operation* is also defined on two models, but these are defined on relations with an identical attribute set. This operation uses distributed data mining (DDM) algorithms to get the results. Another possibility is the use of meta-learner algorithms (e.g. [17]). Requirements are – equivalent to the join operation – identical interestingness function and identical labels in the two models. Furthermore, the models have to be of the same type.

Definition 11. Assuming the models m_1 and m_2 have the same schema $M_1 = M_2 = \{C_R, I, L_1, \dots, L_n\}$ and they are of the same type. The *union operation* $\cup$ is defined as:

$$\begin{aligned} m_1 \cup m_2 &= \{t : t \text{ is defined by a DDM algorithm with} \\ &\quad t[C_R] \in dom(C_R), t[I] \in [0,1], \\ &\quad t[L_1] \in dom(L_1), \dots, t[L_n] \in don(L_n)\} \end{aligned}$$

The resulting schema is $M_1 = M_2 = \alpha(m_1 \cup m_2)$.

4.1.5 Difference

The *difference operation* is another operation, which can be defined as a constraint algebra operation *set difference*. The difference is defined between two models which have the same schema. The different patterns are defined over the union of both relations, so we can compare patterns only by the describing constraint tuple. The set difference returns all patterns whose extensions are not identical. Thus, the result of the model difference are all patterns with interestingness over the union of the mined relations. This operation is reasonable, if the analyst wants to check if a model contains new ideas.

Definition 12. Let m_1 and m_2 be two models with the same schema $\alpha(m_1) = \alpha(m_2) = \{C_R, I, L_1, \dots, L_n\}$ over the relation r_1 and r_2, respectively. The *difference* $-$ is defined as following:

$$\begin{aligned} m_1 - m_2 &= \{t : t \in m_1, t[C_R] \in (m_1[C_R] \overline{\setminus}^s m_2[C_R]), \\ &\quad t[I] = i(t[C_R], r_1 \cup r_2)\}. \end{aligned}$$

The intersection operation can be derived from the difference operation. In this case the intersection is defined as

$$m_1 \cap m_2 = m_1 - (m_1 - m2).$$

This intersection is also a set intersection, that means, all patterns that exist in each of both models are in the result set.

4.1.6 Renaming

The definition of the *renaming operation* $\beta_{B \leftarrow A}$ is straightforward and is based on the renaming operations in constraint databases and relational databases, respectively. The renaming can be applied to the constraint attribute by using the constraint renaming, which replaces all occurrences of A with B in all constraint tuples.

4.2 Operations between Models and Data

Besides the operations of the model algebra the proposed framework includes two operations for transition between the data and model view: the *mining step* which creates a mining model from a relation and the *model-data-join* which gets the extensions of the pattern and applies the properties to the data. The latter operation can also be used to test a predictive model (e.g. a classification) or to predict a new data set with help of a predictive model.

4.2.1 The Mining Step

The mining operation creates a mining model from a relation r. The parameter t of the operation supports the specification of the kind of the mining operation, for instance classification. The resulting model is of the type t and contains patterns over the attribute set of the schema $R = \{A_1, \dots, A_k\}$. The interestingness function and the labels are chosen according to the type of the model and the operation. A following selection on the interestingness values can be used to express some thresholds for the mining algorithms. Thus, the mining algorithm is processed with the selection thresholds during the evaluation of the whole expression.

Definition 13. The *mining operation* μ of type t uses as input a database instance r with the schema R. It creates mining model m of type t over the instance r:

$$m = \mu_t(s).$$

The model m has the schema $M = \{C_R, I, L_1, \dots, L_n\}$.

4.2.2 Model-Data-Join

The transition from the model view to the data view is defined by the *model-data-join* which creates a relation containing the tuples represented of a pattern. The tuples are extended by the labels of the containing pattern. The resulting relation represents the extension of the pattern according the relation to be joined.

As the mining-models can be divided into two groups – predictive and descriptive models – the model-data-join is used for different tasks. There are three scenarios in the case of a predictive mining model, e.g. a classification model: *testing*, *prediction* and *getting the extension*. In the first scenario a test set is applied to the mining model via the model-data-join. In the result a comparison between applied class name and test class name is possible. During the prediction a new data set is joined with the model and new attributes are connected to the tuples. Getting the extensions of the pattern from the training set can be used in the predictive as well as in the descriptive case. Formally, the model-data-join is defined as follows:

Definition 14. Let m be a model with the schema $M = \{C_R, I, L_1, \ldots, L_n\}$ and r_1 be a relation with the schema R_1. Furthermore, it holds $R \cap R_1 \neq \emptyset$. The *model-data-join* $\bowtie_{md}$ is defined as:

$$\begin{aligned} r_2 &= r_1 \bowtie_{md} m \\ &= \{\, t \ : \ t[R_1] \in r_1, \exists p \in m \ : \ t[R_1] \in ext(p[C_R]), \\ &\quad\ t[L_1, \ldots, L_n] = p[L_1, \ldots, L_n]\} \end{aligned}$$

The schema of the resulting relation r_2 called R_2 is defined as $R_2 = \{R_1, L_1, \ldots, L_n\}$.

4.3 Example

After providing operations over models and between models and data the proposed framework is described by means of an example. There are two relations with customer information and one relation with transaction data, which stores the information about the bought items of the different customers. At first the customers are classified in relation $cust_1$ and $cust_2$ with the classification algorithms. This is performed by $m_1 = \mu_{class}(cust_1)$ and $m_2 = \mu_{class}(cust_2)$, respectively. Following, we combine the models by using the *union*-operator, which executes a meta-learning algorithm and creates the model $m_3 = m_1 \cup m_2$. From the resulting mining model m_3 the analyst selects the "highrisk" pattern by using the label selection operator, $m_4 = \sigma_{CL='highrisk'}(m_3)$. The mining model m_4 represents all classification patterns that describe the "highrisk" customers.

The analyst selects all highrisk customers from relation $cust$ by applying a model-data-join $cust_{hr} = m_4 \bowtie_{md} cust$. A join between the transaction relation $trans$ and $cust_{hr}$ returns the transactions of all highrisk customer $trans_{hr}$. Over these the analyst executes a frequent set algorithm to get the typical buying behavior of this customer group. Figure 2 illustrates the whole process.

$$\begin{tikzcd}
\begin{array}{c} cust_1 = \{cid, name, age, sal, \\ education, address, class\} \end{array} \arrow[d, "\mu_{class}(cust_1)"] & \begin{array}{c} cust_2 = \{cid, name, age, sal, \\ education, address, class\} \end{array} \arrow[d, "\mu_{class}(cust_2)"] \\
m_1 = \{C_{cust_1}, I, CL\} \arrow[dr, "m_1 \cup m_2"] & m_2 = \{C_{cust_2}, I, CL\} \arrow[d] \\
& m_3 = \{C_{cust_1}, I, CL\} \arrow[d, "\sigma_{CL='highrisk'}(m_3)"'] \\
\begin{array}{c} cust = \{cid, name, age, income, \\ education, address\} \end{array} \arrow[dr, "cust_1 \bowtie_{md} m_4"] & m_4 = \{C_{cust_1}, I, CL\} \arrow[d] \\
trans = \{cid, It_1, \ldots, It_l\} \arrow[dr, "trans \bowtie cust_{hr}"] & cust_{hr} = \{cust_1, CL\} \arrow[d] \\
& trans_{hr} \arrow[d, "\mu_{freqset}(trans_{hr})"'] \\
& m_5 = \{C_{trans_{hr}}, I_1, L_1\} \arrow[d, "\sigma_{t \subseteq (It_1=1 \wedge It_2=1), I_1 \geq 0.8}(m_5)"'] \\
& m_6 = \{C_{trans_{hr}}, I_1, L_1\}
\end{tikzcd}$$

Figure 2: Example process

5. RELATED WORK

There is a lot of work done in optimizing and scaling data mining algorithms. Overviews over this work are for instance [15] on classification and [5] on clustering. Association rule mining was introduced in [2].

Very close to the work presented in this paper is the *3W model and algebra* proposed in [11]. In this work a 3 world model is introduced. The intensional world describes the mining result. Hereby, hierarchical constraint attributes are used. The central objects are dimensions, which are sets of related regions. Over these objects a dimension algebra is defined. Using bridge operators the transitions between the different object classes are secured. In the framework of this paper additionally the notion of the interestingness function is introduced to provide information about quality of the mining results. Furthermore, the proposed framework relies on collective and distributed data mining algorithms. Collective data mining is proposed in [13]. The work discusses algorithms for regression and decision tree induction. [17] give an overview about techniques and issues of distributed data mining and shows an approach of a meta-learning system.

A second work in this area are the *inductive databases* proposed in [9, 4]. In these works the KDD process is modeled as a sequence of queries on inductive databases. An inductive databases consists of a relational database and patterns over this database. A semantic function w.r.t. the data is assigned to the patterns. In these works only association rules and similar rules are considered. Furthermore, no unifying concept for describing patterns is provided.

Several data mining languages or SQL extensions are proposed in literature. Examples are DMQL [8], or MSQL [10].

There exists several standards for data mining. OLE/DB for Data Mining [16] defines data mining models as first class objects and a data mining language which features also a prediction join. PMML [1] is a XML-language for describing data mining models. It defines the syntax for different pattern types. Furthermore, SQL/MM Part 6 Data Mining defines a standardized SQL interface for mining algorithms, that consists of different user-defined types and functions.

The model of constraint database systems was introduced in [12, 19]. [14] gives an overview over many aspects of constraint databases. The relational algebra was extended to use in constraint databases in [7, 19]. An extended constraint algebra was introduced in [3]. In this work the authors propose a nested relational interpretation of constraint relations which is used in the framework of this paper. Furthermore, the authors introduce external functions to constraint databases.

The use of constraint databases in the learning and data mining context was described in [18]. The authors propose a ILP (Inductive Logic Programming) system which learns constraints over symbolic and numeric facts by means of constraint databases.

6. CONCLUSION AND FUTURE WORK

In this paper a vision of a framework for KDD processes is shown. The model consists of two parts: the model view and the data view. The data view is defined in the relational model, the model view has to support *mining models*, for which operations were defined. Mining models contain all information about the results of data mining techniques and provide a uniform description of the results. The defined operators enable powerful querying and manipulation operations.

Future work addresses the implementation of the framework, finding of rewrite rules as well as the support of models with optimized algorithms and index techniques like R-trees or multidimensional hashing. Based on the model and data view a process view has to be defined which supports iterations and conditions as well as uniform visualization possibilities. Another research direction is to find a uniform interestingness function.

7. ACKNOWLEDGMENTS

The author wants to thank his colleague *Oliver Dunemann* for his help during the preparation of this paper.

8. REFERENCES

[1] PMML 1.1 – Predictive Model Markup Language. http://www.dmg.org/html/pmml_v1_1.html, March 2001.

[2] R. Agrawal, T. Imielinski, and A. N. Swami. Mining Association Rules between Sets of Items in Large Databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the ACM SIGMOD 1993, Washington, D.C.*, pages 207–216. ACM Press, 1993.

[3] A. Belussi, E. Bertino, and B. Catania. An Extended Algebra for Constraint Databases. *TKDE*, 10(5):686–705, 1998.

[4] J.-F. Boulicaut, M. Klemettinen, and H. Mannila. Modeling KDD Processes within the Inductive Database Framework. In *Proceedings of DaWak99, Florenz*, pages 293–302. Springer-Verlag, 1999.

[5] D. Fasulo. An Analysis of Recent Work on Clustering Algorithms. Technical Report 01-03-02, University of Washington, April 1999.

[6] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery: An Overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 1, pages 1–34. AAAI Press/ The MIT Press, 1996.

[7] D. Q. Goldin and P. C. Kanellakis. Constraint Query Algebras. *Constraints*, 1(1/2):45–83, September 1996.

[8] J. Han, Y. Fu, K. Koperski, W. Wang, and O. Zaiane. DMQL: A Data Mining Query Language for Relational Databases. In *Proceedings DMKD'96, Montreal, Canada*, June 1996.

[9] T. Imielinski and H. Mannila. A Database Perspective on Knowledge Discovery. *Communication of ACM*, 39:58–64, 1996.

[10] T. Imielinski and A. Virmani. MSQL: A Query Language for Database Mining. *Data Mining and Knowledge Discovery*, 3(4):373–408, December 1999.

[11] T. Johnson, L. V. Lakshmanan, and R. T. Ng. The 3W Modell and Algebra for Unified Data Mining. In *Proceeding of the 26th VLDB Conference, Cario, Egypt*, 2000.

[12] P. C. Kanellakis, G. M. Kuper, and P. Revesz. Constraint Query Languages. *JCSS*, 51(1):26–52, August 1995.

[13] H. Kargupta, B.-H. Park, D. Hershberger, and E. Johnson. Collective Data Mining: A New Perspective toward Distributed Data Mining. In H. Kargupta and P. Chan, editors, *Advances in Distributed and Parallel Knowledge Discovery*, chapter 5, pages 131–178. MIT/AAAI Press, 2000.

[14] G. Kuper, L. Libkin, and J. Paradaens, editors. *Constraint Databases.* Spring-Verlag, Berlin Heidelberg, 2000.

[15] T.-S. Lim and Y.-S. Shih. A Comparision of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms. *Machine Learning*, 40(3):203–229, September 2000.

[16] A. Netz, S. Chaudhuri, U. M. Fayyad, and J. Bernhardt. Integrating Data Mining with SQL Databases: OLE DB for Data Mining. In *Proceedings of the ICDE 2001, Heidelberg, Germany.*, pages 379–387. IEEE Computer Society, 2001.

[17] A. L. Prodromidis, P. K. Chan, and S. J. Stolfo. Meta-Learning in Distributed Data Mining Systems: Issues and Approaches. In H. Kargupta and P. Chan, editors, *Advances in Distributed and Parallel Knowledge Discovery*, chapter 3, pages 79–112. MIT/AAAI Press, 2000.

[18] T. Turmeaux and C. Vrain. Learning in Constraint Databases. In S. Arikawa and K. Furukawa, editors, *DS '99, Tokyo, Japan*, volume 1721 of *LNCS*, pages 196–207. Springer, 1999.

[19] J. van den Bussche. Constraint Databases, Queries, and Query Languages. In Kuper et al. [14].