CWI Quarterly Volume 12 (1) 1999, pp. 1 – 21

Experiences with Knowledge Discovery Paradigms

Willi Klösgen
German National Research Center for Information Technology (GMD), D-53757 St. Augustin
e-mail kloesgen@gmd.de

Knowledge discovery in databases can informally be introduced as large scale search for interesting patterns in databases. So the main paradigms that have to be operationalized in a KDD system are patterns, search, and evaluation of interestingness. In this paper, we summarize the approaches implemented in the discovery system Explora to realize these paradigms and discuss experiences and problems.

1. Introduction: The KDD Process and related Paradigms

Knowledge discovery is a data analysis process. But compared to conventional "manual" analysis, a discovery process is supported by a KDD system established with a high degree of autonomy, especially with the ability to construct, process and evaluate large search spaces of hypotheses. However, a discovery process mostly cannot be specified in advance and automated completely, because it depends on dynamic, result dependent goals and intuitions of the analyst and emerges iteratively. Typically, a process consists of many steps, each attempting at the completion of a particular discovery task, and accomplished by a targeted application of a discovery method, i.e. a discovery strategy. The process iterates many times through the same application domain, based on search in various hypotheses spaces. Therefore, an appropriate synergy of the user and the KDD system has to be provided.

Mining for instances of different types of patterns (rules, changes, trends, etc.) is the central task within this process. However, in the whole process of obtaining knowledge, inference of patterns is only one and often small part. The whole process starts with data collection and cleaning tasks including selecting, combining, and deriving data as well as providing domain knowledge to be exploited by the system. Then mining tasks and appropriate methods and their parameters are selected according to the types and application purposes of knowledge that shall be discovered and the problem and data types. The major data mining tasks are classification, clustering, dependency modeling, summarization, change and trend detection, and interactive visualization to explore e.g. space and time related flows. After the pattern instances have been derived in the data mining step, postprocessing of the found patterns includes selecting and constructing elaborated, truly interesting patterns and their presentation. Finally, the discovered knowledge is used.

When identifying the main paradigms that are used as general approaches in KDD and that differ from conventional data analysis approaches, one can concentrate on the mining and postprocessing phases. The preprocessing tasks are mostly commonplace in any data analysis work. They include data exploration to get a first feeling of the domain and data, supported for instance by interactive graphical presentations (dynamic scatter plots and other interactive statistical presentations, parallel coordinates, netmaps, etc.) and data preparation by cleaning, derivation and reduction techniques, and the selection of an appropriate data model. Especially when datasets are very large or several operational databases have to be combined, also technical aspects of datamanagement play an important role in preprocessing, including ensuring efficient data access possibly achieved by a datawarehouse or an own datamining database relying on an inverted organization of data.

A KDD specific task within preprocessing is the specification of domain knowledge that can be exploited in the following mining and refinement phases. The role of domain knowledge in KDD is studied below. Another important point is data reduction, primarily addressing the often very large number of variables in large scale datamining applications. This can be done statically in a preprocessing step, e.g. by feature selection techniques, or, as will be discussed below, dynamically during a mining task.

In the following subsections, we introduce the main paradigms that are characteristic for data mining and hypotheses evaluation and refinement.

1.1. The pattern paradigm

The central paradigm for data mining is that of a pattern. In most informal definitions, KDD is introduced as the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data (Fayyad et al. [2]). A pattern is seen as a class, type or schema of a statement, and an *instance* of a pattern is a concrete statement S in a high level language that describes a subset $D(S)$ of a database D with a quality $q(S)$ (Frawley et al. [4]; Klösgen & Zytkow [12]). Thus, from a data analysis or statistical viewpoint, a pattern can also be seen as a model representing a subdomain of the application domain in a formal language or as an hypothesis or assumption that has to be statistically verified in data. Constitutive for a pattern class are the meaning, i.e. the statistical content or problem type of the abstract pattern type or model class, the verification and evaluation methods testing an hypothesis associated to a pattern instance and measuring its quality by operationalizing several aspects of interestingness, the presentation schemes, the arguments or search dimensions, and the search characteristics and search control properties. For a pattern type, there are several options for verification and quality computation, presentation in natural language or graphical form, and search control.

A pattern instance is treated by a discovery system as a node in a search space which is processed by a search algorithm, an hypothesis on a regularity in data, tested and evaluated by a verification and a quality method, and a finding which is presented to the user.

1.2. The search paradigm

Therefore, large scale and mostly brute force search is another important KDD paradigm. Search is the core of most data mining algorithms. In a sense, the search approach mimics the procedure of data analysts when looking at a set of cross-tabulations to find interesting cells or when generating a sequence of statistical (regression) models. Compared to the limited manual analysis capacities, a computerized brute force search can be organized more systematically and more completely to cover large parts of hypotheses spaces. Therefore, already in the statistics and data analyses community, there were early implementations of this brute force paradigm. Statistical packages like SPSS allow to generate all cross- tabulations, offer heuristical approaches searching for good regression models by applying forward selection, backward elimination and stepwise strategies, and CART methods (Breiman et al. [1]) and their predecessors generate classification and regression trees.

However, most early brute force techniques in data analysis had a bad reputation among statisticians. They were often characterized as "torture the data until they confess" and a main critique referred to testing many hypotheses leading necessarily to results that will occur just by chance. Especially the last point has to be observed very seriously in datamining methods: Bonferroni adjustment or similar techniques to adapt the test criteria to the huge number of tests, and applying independent training and test datasets are necessary to ensure statistically valid results.

Search strategies can be optimizing or satisfying. Optimizing strategies aspire to discover pattern instances (hypotheses) with the best quality, satisfying strategies derive all instances that satisfy the given constraints. Search can be exhaustive or heuristic. An exhaustive strategy prunes only hypotheses that cannot belong to the solutions, whereas heuristic strategies like hill-climbing and its extensions (tabu search, simulated annealing), beam-, tree-, or stepwise search aspire to process prospective regions of the search space, but cannot exclude that there are (better) solutions in the pruned subspaces. Webb [18] proposes an efficient exhaustive search strategy; but often the search space cannot be restricted in a way that allows exhaustive search (e.g. by limiting the order of conjunctions, applying only coarse discretizations and taxonomies). Another possibility to avoid combinatorial explosion is to apply constraints on the search space. Ideally, such constraints represent domain knowledge, preventing also from discovering many uninteresting pattern instances.

Search can be performed automatically in large hypotheses spaces or under an iterative and detailed control of the user selecting at each iteration step a prospective subspace of the overall search space and, stimulated by the results in this subspace, directs the search to a next subspace. The selection of subspaces is influenced by the evaluation of interestingness of the hypotheses in a subspace.

1.3. The interestingness paradigm

Having introduced verification and qualities for pattern instances, we mainly addressed the statistical evaluation of interestingness of the associated hypotheses. The statistical validity of a hypothesis includes the aspects *strength of a pattern* (deviation of a mean in a subgroup from the overall mean, probability or certainty of a rule, etc.) and *generality* of the pattern (size of the subgroup). *Simplicity* refers to the syntactical complexity of the presentation of a pattern. *Novelty* and *usefulness* are more subjective aspects of interestingness which are not yet implemented in most current KDD systems. Some proposals to treat *novelty* can be based on adaptive behaviour of the system learning from the user when selecting among presented findings. MATHEUS ET AL. [16] treat the *usefulness* facet: In the KEFIR system, the *utility* of a finding is determined based on its estimated benefit from a possible action connected to it. MAJOR and MANGANO [13] describe a semiautomatic process of identifying interesting rules among a vast amount of rules.

One technique to deal with interestingness is based on constraining the search space. Domain related constraints are defined that are used to exclude non-interesting patterns. In an association rule application, e.g. when searching sets of products that are typically jointly bought in mail-orders, associations between products that appear on the same page of the catalogue issued by the mail order company may be defined as uninteresting by specifying a corresponding constraint between products. Other search space constraining techniques are applied when syntactical biases are defined for the hypotheses language (like a conjunctive description language for subgroups), e.g. by various preference specifications related to groups of variables. Similar variables are arranged in a group and conditions are specified on how many variables are to be considered from each group in what order.

A special problem area for operationalizing interestingness in KDD is the integration of several aspects of interestingness. Several integration types can be distinguished. A first possibility is given by integrating various interestingness aspects into a single one-dimensional result, e.g. by a weighted sum of the single aspects, or by an evaluation function combining several measures like the product of the square root of the subgroup size and the subgroup strength (as usually done in statistical evaluations). Another possibility is a multidimensional interestingness result. Then multidimensional orderings of interestingness vectors are studied, or the the single aspects are ordered hierarchically (MICHALSKI [14]).

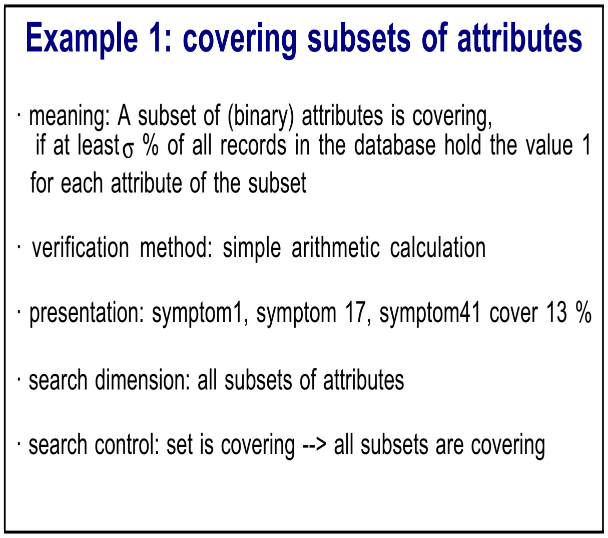
In the following sections, we will present the main realizations in the discovery system Explora (Klösgen [9, 10, 11]) that relate to these paradigms and some of the experiences with these solutions.

2. Patterns in Explora

Since the pattern paradigm incorporates a very broad concept, many mining tasks can be captured within this framework. Figure 1 gives some first simple, but not KDD-typical examples implemented in Explora.

2.1. Types of data analysis problems represented with Explora patterns

A simple classification of data analysis questions or problem types is first organized along the dichotomy of whether any dependent variables shall be analyzed with respect to their relations to independent variables. This leads to the main groups of identifying dependencies between variables (also called supervised learning in Machine Learning terminology) and clustering approaches (unsupervised learning). Explora does not include clustering patterns, but only some simple patterns for unsupervised discovery (e.g. the patterns illustrated in Figure 1a and 1b).



Example 1: covering subsets of attributes

· meaning: A subset of (binary) attributes is covering, if at least σ % of all records in the database hold the value 1 for each attribute of the subset

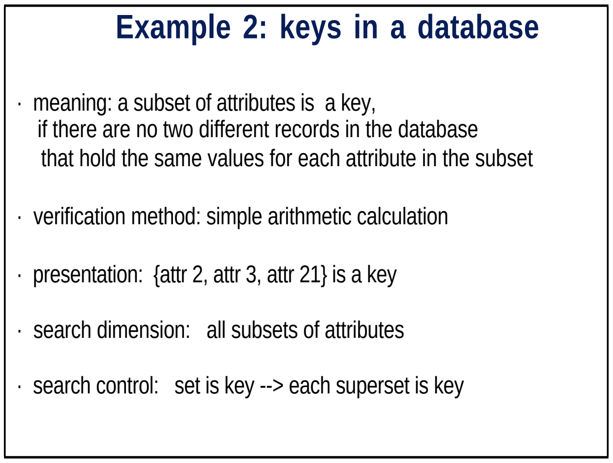
· verification method: simple arithmetic calculation

· presentation: symptom1, symptom 17, symptom41 cover 13 %

· search dimension: all subsets of attributes

· search control: set is covering --> all subsets are covering

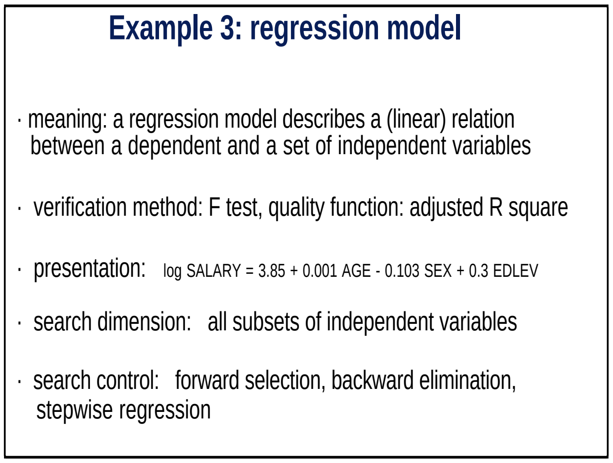
Figure 1a. The components of the pattern *covering subsets of attributes*



Example 2: keys in a database

- meaning: a subset of attributes is a key, if there are no two different records in the database that hold the same values for each attribute in the subset
- verification method: simple arithmetic calculation
- presentation: {attr 2, attr 3, attr 21} is a key
- search dimension: all subsets of attributes
- search control: set is key --> each superset is key

Figure 1b. The components of the pattern *keys in a database*



Example 3: regression model

- meaning: a regression model describes a (linear) relation between a dependent and a set of independent variables
- verification method: F test, quality function: adjusted R square
- presentation: log SALARY = 3.85 + 0.001 AGE - 0.103 SEX + 0.3 EDLEV
- search dimension: all subsets of independent variables
- search control: forward selection, backward elimination, stepwise regression

Figure 1c. The components of the pattern *regression model*

One reason for the absence of conceptual clustering approaches in Explora is partially demand-driven, possibly by a biased selection of application domains where the analysis of dependent variables was required. But another reason is the implementation of the general search approach in Explora which is not so fundamental for clustering techniques. Many conceptual clustering approaches are realized by incremental or constructive techniques. The Cobweb system (Fisher [3]), for instance, incrementally assigns a next object to a class in the current clustering or performs such operations as creating a new class, merging and splitting classes. Cluster/2 (Michalski & Stepp [15]) constructs at first stars from seeds and then a disjoint cover for stars. So search in hypothesis spaces is not the appropriate paradigm of these techniques.

The predominant part of the patterns implemented in Explora refer to the analysis of one or several dependent variables. These dependent variables which are selected by the user are studied for their relations to independent variables (also selected by the user). KDD usually deals with given databases that collect data on objects or cases such as persons. Subgroups of these objects are constructed referring to the independent variables (e.g. males, high income persons over 65 years old). Then the dependent variables are analysed in these subgroups by evaluating a distributional parameter (mean, share, determinant of covariance matrix, etc.) of the dependent variables in the subgroup.

A classification of pattern types for analysing dependencies can be arranged along the dimensions of *type of dependent variables* and *number of population* to be analysed. This corresponds to the usual classification of statistical problems, e.g. in the context of selecting statistical tests.

When analysing a single population, the distributional parameter of the dependent variables in the subgroup is compared with the value of the parameter in the whole population. For the analysis of several populations, the values of the parameter in the subgroup of these populations are compared.

By providing these various pattern types in Explora, a broad range of analysis problems can be treated, ranging from rules for identifying deviations in subgroups from some overall behaviour, to identifications of changes and time trends when analysing regularly collected data. Patterns for both categorical and continous dependent variables can be discovered. The next step would be to analyse co-occurrence patterns for the results found in these different hypotheses spaces. For instance, when a subgroup was identified that changed its behaviour between two time points, one can analyse the behaviour of the subgroup in other hypotheses spaces and derive explanations for the change.

2.2. Verification and quality methods

The verification and quality functions that are used to evaluate the subgroups refer to diverse statistical problem types. As mentioned in 2.1, one of various distribution parameters of dependent variables in the subgroup determines the meaning or problem type. Figure 2 lists some verification methods within a classification of problem subtypes according to the two dimensions "type of dependent variable" and "number of populations compared" that are used in Explora to offer some 30 pattern types for capturing interestingness of subgroups.

A verification method is associated to a statistical test for a distribution parameter, e.g. a mean. Then a mean test verifies that the mean of the dependent variable in the subgroup differs significantly from an a-priori value like the overall mean in the population, or that the mean of the dependent variable in the subgroup is significantly higher in a first population than in a second population, or that the mean of the dependent variable in a subgroup

type dep. var	1 population	2 populations	*k* populations
binary	binomial test chi 2 ITRULE	hypergeometric distribution chi 2	chi 2 heuristic tests
nominal	INFERULE chi 2, CN2	chi 2	chi 2 heuristic tests
ordinal	median test	comparison of medians	
continuous	mean test	comparison of means	

Figure 2. A classification of pattern subtypes in Explora

strongly increases over time (when a population of cases is given for each point in a series of time points).

Statistical tests in the verification methods compute evidence (mainly statistical significance). For exploration, we need no exact and elaborate statistical tests, but simple approximations, because we perform many tests in a large hypotheses space. A too small significance threshold leads to many findings, some of which are statistically not justified, because they are merely random results. Moreover, a large number of findings will confuse the user and give him the possibility to select perhaps those statements confirming his prejudices. Generally, we therefore select large significance thresholds. Using an explorative test, we decide, whether a distinct value of a random variable (the number of target elements in a subgroup, the mean of a target variable in a subgroup, etc.) differs from an expected value significantly. We calculate the expected value and (mostly an estimation of) the variance of the random variable. For an exact test, we need the distribution function of the random variable, but for exploration, we can use approximations. In our explorative tests, we use the expected value and the variance, and as thresholds for regarding a value as significant, deviations of the value from the expected value that exceed a fixed multiple of the standard deviation.

We use only deviations from the expected value of at least $3s$, where s is mostly an estimation of the unknown standard deviation. Furthermore, we regard patterns with deviations between $3s$ and $5s$ only as weak indications for a significant finding. For a normal distributed random variable, the probability of deviations of more than $2s$ is about 5% and of more than 3s is about 0.3%. This is also true for other distributions (t-, binomial, hypergeometric distribution) which are used in Explora tests (Klösgen [8]). Therefore, a level of significance of 5 is relatively high, but takes into account the multitude of tests and the approximations (no normal distributions, no independencies, noise, etc.).

Such a verification method is introduced first to exclude random results.

Since the hypotheses spaces in Explora are restricted to some tenthousands of hypotheses (by space and run time constraints of the software implementation), the above selection of the $5s$ condition is reasonable. Further, this measure (deviation of observed value of parameter from expected value related to its standard deviation) is used to measure the quality of a pattern instance. For example, in the two cases of probabilistic rules and of comparing the share of a target group in a subgroup for two populations, the following quality functions are derived from this measure:

$$Q(p,g) = sqrt(g)^{*}(p - p0) \tag{1}$$

$$Q(p_1, p_2, g_1, g_2) = sqrt(g_1^{*} g_2 / g^{**}2)/sqrt(p^{*}(1-p))^{*} sqrt(g)(p_1 - p_2) \tag{2}$$

g and p refer in (2) to the union of the two populations

These functions combine the generality of a pattern (size g of the subgroup) with the strength of a pattern. The influence of the size of a subgroup is measured by its square root, and the strength by the difference of the conditional probabilities (share of target elements in subgroup).

Other options of quality functions for these two and other patterns are offered in the Explora system. Appropriate choices depend on the discovery goals, e.g. accuracy of single descriptions versus optimal overall set of descriptions. In (KLÖSGEN [11]), some contributions to a theory of quality functions are presented.

Experiences with different quality functions refer to the two aspects of ensuring statistical significance (verification test) and calculating a numerical quality for an hypothesis. Especially, when databases are large (e.g. more than tenthousand cases), the selected statistical test is often not so important. In these cases, many hypotheses are validated with a significance level of 10 or higher. Thus, it plays no role for most of the selected hypotheses, if their significance level is calculated as 9.5 or 10.5 by using a different statistical test, when a minimal level is reasonably set to 5.

Quality functions must be selected in the context of search strategies. When a simple satisfying approach is used and all verified hypotheses are presented in the order of their qualities, then also the effect of different quality functions is not so outstanding. By using a different quality function, typically nearly the same hypotheses are presented, however in a different ordering. When selecting a quality function that highly emphasizes the size of a subgroup, more general subgroups will be presented first, even if the stength of the pattern in the subgroup is not so high.

The situation is different when refinement methods based on a suppressing algorithmus are used (compare Section 3). Here the hypothesis with a higher quality has a larger potential to suppress another similar hypothesis. As one rule to select between quality functions, we use the direction of the user, whether the generality criterium has a higher preference for the analysis problem than the strength of the pattern. The square-root in the functions (1) and (2) can then be replaced by some other exponent for adapting the quality function to the required properties of resulting hypotheses. More expert-knowledge has to be identified and incorporated in future KDD systems which is related to providing different quality functions for a pattern type and to install expert selection rules based on global directions of the user about the general properties the resulting findings shall have.

2.3. Search dimensions

A pattern has been introduced as a statement about a subset of the data. To address a subset, Explora uses the following concepts. A *population* refers to a segmentation of the database that is derived by the values of a distinguished variable. Typically this variable represents time and a population then includes all the cases belonging to a selected time point in a database that includes cases for different time points. Another frequently distinguished variable represents countries, then a population includes all the cases of a special country in a multi-national database.

A *range* is a subset of the cases of a population that is described and selected by a selection query (typically we use conjunctions of propositional expressions as selection queries in Explora). Ranges are introduced to further restrict the statements on subsets of the populations. A space of ranges is defined by selecting some variables and further details for the description language such as to use at most k conjunctions in a range description. A range space is a set of descriptions which is partially ordered by generality.

An *variable* set is a selection of variables and is mostly used for selecting dependent variable(s) for dependency patterns (its use in the coverage and key patterns 1.1 and 1.2 is obvious). For dependency patterns, this can either be a set of categorical variables (or discretized continous variables) or a set of continuous variables. For continous variables, a partial ordering is defined in a variable set as the subset relation. In case of categorical variables, a partially ordered set of subsets of a population (target sets) is defined by a variable set.

A *subgroup* is also a selection of cases of a population which is syntactically built like a range or a target group. Subgroups are constructed by selecting the independent variables for a dependency pattern. Therefore syntactically there is no difference between a range, a target group and a subgroup. A range represents a restriction, a target group the dependent part and a subgroup the independent part in a dependency pattern.

The graphical user interface of Explora allows the specification of these three search dimensions, each to be constructed as a partially ordered space, by selecting variables and their values (including taxonomies) in a range, a variable set (dependent variables) and an independent variables window. By selecting one or several populations and the type of dependent variables, implicitely the pattern type is selected (according to the classification of pattern types by number of populations and type of dependent variables). The Explora search space is then constructed as the product space of these partially ordered spaces, where a product partial ordering is implied. For details see (KLÖSGEN[10]) where also the internal search layer of Explora is described. The internal search layer of Explora applies to the product of an arbitrary number of any partially ordered spaces, but the Explora GUI is restricted to the search dimensions characterized above.

2.4. Search control

Search control directions are prespecified when implementing a pattern type in Explora and can be partially modified by the user. There are two main controls referring to the hierarchical ordering of search dimensions and to redundancy filters. The default hierarchical ordering of processing and presenting search spaces and their results is top-down and proceeding from ranges as the outer loop, over dependent subsets like target groups to the independent subgroups as the inner loop. Thus, at first subgroups are found for the most general range(s) and the most general target group(s). Then, for the most general range(s), subgroups are found for more special target groups. Depending on the type of the pattern, typically at first the most general subgroups are analysed and then their specializations.

This results of course in a large number of findings, even if a single inner loop on subgroups and their refinement only produces a moderately sized set of results. Therefore, the heuristic pruning criterium "if true, then successor not interesting" is applied. If a finding can be verified for a range, a target group and a subgroup, then the space of hypotheses that refer to more special ranges, target groups, and subgroups is pruned.

Other filters of the type "if true, then successor true" or "if true, then predecessor true" can be exploited for special pattern types (see figures 1.1 and 1.2) and for special constraints. If the filters are specified for special constraints, then the pruning flags are set within the verification method, otherwise they are associated to the verification result. The coverage constraint, e.g., requiring a minimal number of target elements in a subgroup and a range, infers for all three search dimensions the redundancy filter "if true, then predecessor true". This means, if a triple consisting of a range, a target group, and a subgroup does not satisfy this constraint, then also any specialization will not satisfy the constraint and can be pruned from further search.

The redundancy filters proved as a powerful feature to reduce the search effort, and for one-dimensional search spaces, the number of presented results (search space of subgroups with fixed target group and range). However, in the case of multi dimensional search spaces with large spaces of ranges and target groups, often the number of presented results is still very large, so that additional solutions have to be found. More interactivity involving the user in influencing the running search process could be a prospective solution, but was not implemented in Explora.

2.5. Presentation methods

Findings are presented in natural language form. Text templates can be provided by the user for an hierarchical arrangement (see 2.4) of the search dimensions of a pattern type. Header templates typically relate to the range and target group hierarchies, and detail templates to the subgroup loop (which usually is the inner loop). Open positions within these templates are provided for the dimensions of a pattern type. These open positions are replaced by a natural language representation of an element of the component search dimension, while the remaining text specified in the template is fixed. A method has to be provided that generates the natural language representation for an element of a search dimension, e.g. for a subset of cases representing a selection query defined by a conjunctional expression (compare subsection 3). Figure 3 shows an example of results presented on the basis of text templates.

By modifying these templates, the user has some flexibility in adapting the result presentation to a special application domain. However, the presentation of results is basically static, i.e. the results are presented as a hierarchical list of findings offering to the user no possibility to operate actively on the results, e.g. by selecting individual findings and starting a new search or refinement task related to the selected findings.

An experimental implementation of graphical result presentation has been added (Klösgen [8]). A finding selected by the user is illustrated by a traditional presentation graphic (bar chart, pie chart, etc.), where some graphical design knowledge is coded in expert rules selecting and compiling an appropriate type of graphical presentation. More advanced presentation methods should provide additional statistical information such as confidence intervals (e.g. fourfold display representing a rule and the confidence intervals for the probabilities associated to a rule) and graphical methods to present not only single results, but a result space (e.g. the rule and tree navigators developed by Silicon Graphics).

3. Performing Search in Explora

The search layer of Explora provides an environment supporting search tasks in a product space of partially ordered sets. Search is scheduled in two phases: a basic brute force search and a refinement phase. Basic brute force search is an exhaustive search. To overcome the time complexity of exhaustive search, pruning conditions are included based on redundancy filters (see 2.4). Additionally, some heuristic strategies are offered that are composed from basic search steps in subspaces, e.g. a stepwise search approach iteratively selecting an appropriate subset of variables to be used for subgroup construction, a tree and a beam search approach. Subspaces constructed by these heuristical approaches have a limited search depth (i.e. depth = 1). Figure 4 summarizes the basic brute force search.

In the algorithm of figure 4, the stronger and weaker direction for each search dimension, which can be either successor, predecessor, or not existing,

```
Problem:      Conditions for good productions
              target group: QKG1 < 0.16,  QKG3 < 0.41
Pattern:      Probabilistic rules
Strategy: High accuracy, Small overlapping, Recursive exhaustive search
Productions:  Plant A, 1995
              20% of the productions are good

Subgroups describing the target group:

       66% of PG2L8 = 0, PG4 < 0.18
       94% of PG2L8 = 0, PG4 < 0.18, PG15 > 0.88,  PG15L8 > 0.88
      100% of PG2L8 = 0, PG4 < 0.18, PG1  = 0.25,  PG12 0.450.55
22% coverage of the target group, 9.8% overlapping

       55% of PG8 = 0.7, PG11L8 0.8-0.85
      100% of PG8 = 0.7, PG11L8 0.8-0.85, PG4 < 0.18, PG15L8 > 0.88
      100% of PG8 = 0.7, PG11L8 0.8-0.85, PG1L4 = 0.25
19% coverage of the target group, 5.4% overlapping

       63% of PG1 = 0.5, PG16 0.5-0.6
       90% of PG1 = 0.5, PG16 0.5-0.6, PG15 0.6-0.7, PG15L4 0.6-0.7
       86% of PG1 = 0.5, PG16 0.5-0.6, PG8L8 = 0.75, PG15L4 0.6-0.7
35% coverage of the target group, 9.4% overlapping

total coverage of target group: 66%
total overlapping: 19%
```

FIGURE 3. Conditions for good products

is determined by the redundancy filters specified for a dimension. Stronger nodes can be eliminated, because they are false, and weaker nodes, because they are not interesting. Details are described in (KLÖSGEN [10]). Since the search space structure is built statically, elimination can easily be implemented (see below).

The refinement phase in Explora aims at alleviating the deficiencies of brute force search, especially to overcome the flood of information consisting of too many valid hypotheses by selecting or deriving the truly interesting hypotheses. More general refinement tasks that are not supported in Explora deal with interpretation and visualization, checking and consolidation of conflicts with existing knowledge and the incorporation of the derived knowledge into the application component of a domain. To overcome the problems of brute force search, good results have been achieved in the Explora system with a suppression algorithm (GEBHARDT [5], KLÖSGEN [11]), evaluating both the quality and the similarity of hypotheses. The degree of similarity between two hypotheses controls the suppression of a hypothesis with a weaker quality by a better quality hypothesis.

```
loop over nodes in product search space, following lexicographic ordering:

if logical-value-of-verification-method-of-pattern(node) = true
search for snode::= one strongest true node among nodes stronger than node
optionally treat weaker nodes than snode
insert snode in set of elements of result structure present snode on output
stream (use presentation-template of pattern)
eliminate weaker nodes than snode
eliminate stronger nodes than snode

if logical-value-of-verification-method-of-pattern(node) = false
eliminate stronger nodes

node = next available node (relative to node and lexicographic ordering)
```

FIGURE 4. Basic search loop in Explora

Though basic search relying on pruning of search spaces implied by pruning criteria is useful for discovery of true and partially interesting findings, some aspects of interestingness have to be treated in a refinement phase to overcome diverse problems. A few problems are illustrated now in more detail.

First, consider the trade-off between the strength and the generality of a finding. Regard two hypotheses on a group "Males" and a subgroup "Males, older than 60". Both findings may be significant, but the finding on the subgroup may be more significant. Basic search may only present the group "Males" and then prune the search subspace under this node. A refinement technique decides how much higher the significance of the subgroup must be in order to also (only) present the subgroup.

Refinement of findings also means that heterogeneities within groups of cases are controlled depending on the structures available in the language used to describe groups. For instance, if intervals of the values of a variable (e.g. numerical variable age) are formed to describe groups, one has to exclude a finding about the positive deviation in a significant subgroup "age 30-60", when the subgroup 50-60 shows a negative deviation. The subgroup verified by basic search is significant, but the real cause may be a subset which is highly significant while its complement with respect to the subgroup is more or less insignificant.

Selection techniques can be applied in the refinement phase to save the user from getting overwhelmed with many similar findings where the subgroups are not subsets of one another, such as "Income> 3000, Age>30" and Income>3500, Age>25". The suppression procedure implemented in Explora solves the interval-problem for the one-dimensional case, whereas for multi-dimensional intervals, further techniques must be added to treat some cases of overlapping.

Another problem is due to hidden dependencies between independent variables. Consider the finding "*Mean of Salary is significantly above the average for White Males with Educational Level>16*". Because of correlations between the fields Sex, Minority Status, Educational Level, Employment Category, the same finding also holds for *High Employment Category.* A refinement technique must treat a dependency *White Males, Educational Level*$>$ 16 $\longrightarrow$ *High Employment Category* and decide if there is an additional effect of *Males justifying the finding "Mean of Salary is significantly above the average for Males, High Employment Category".*

These problems are solved by the suppression procedure that employs two notions: a measure of the quality of a single finding and an asymmetric measure of the similarity of two findings. These two cooperate in suppressing findings that are worse than, but similar to other findings. Characteristics of this selection procedure are studied in detail in (GEBHARDT [5]).

The main drawbacks of the Explora search approach are the main memory implementation of the search space and the static construction of individual search dimensions. Search spaces are implemented in an object oriented way, but the objects are maintained in main memory. This restricts the size of the search spaces that can jointly be exploited in a search procedure to approximately 100.000 nodes in the product spaces. This limitation is of course influenced by the available main memory (Explora can be run on Macintosh, and this limitation refers to a comfortable equipment of 32 Megabytes). Another factor which competes for main memory is the size of the database, or more precisely, the size of the currently active data given by the segments and variables selected for a discovery task, which is also held in main memory. Therefore, Explora can be used for medium sized datasets of upto approximately 100.000 cases. Large search spaces of 100.000 nodes will require some 10 hours of performance, small search spaces including only five to ten variables or constructed by heuristic approaches need some minutes. Given the Macintosh environment, it is reasonable not to address any high volume data mining applications, so that the limits of approximately 100.000 cases and 100 variables, and of search spaces with some tenthousands of nodes are practicable. So alltogether, this main memory foundation proves as not such restrictive considering this equipment and data mining environment.

A more serious drawback is the static construction of the search space. This means, that at first the component search dimensions are built as partially ordered spaces, then the product space with the implied ordering, and then search is performed using the statically constructed structure. The system applies operators to construct search dimensions, i.e. partially ordered spaces of objects. One of these operators constructs a partially ordered space of subgroup objects given a selection of independent variables and language parameters such as the maximal number of conjunctions. Based on an object oriented realization, methods are additionally included to provide the meaning of an object of the constructed search dimension (i.e. a bit vector representing a subgroup, provided by the datamanagement interface) and the natural language representation of an object (to be inserted into the open positions of the text templates). The static construction was chosen, because it proved as more time efficient when realizing pruning allowing look ahead elimination of pruned search space elements. Experiments with a dynamic construction of search spaces consumed too much time for checking pruning conditions when dynamically generating a new element of the search space.

However, static construction needs more space, and more important, dynamic discretization of continous variables has not been implemented. The Explora user has only the possibility to statically construct discrete taxonomies for continous variables before starting a search task. By elaborating the search strategies composed of several basic search and refinement steps on restricted partial search spaces, some of the limitations of static construction of (partial) search spaces could be overcome.

4. Dealing with Interestingness in Explora

Since in addition to various application perspectives of a domain, there are so many different pattern types, and for each pattern type, typically an abundance of statistically valid pattern instances, interestingness cannot be handled totally automatically by a discovery system. The user has to give some directions to the system indicating on a relatively high level the current focus of interestingness. So it is not possible just to give the command: "Find me everything interesting in this dataset".

To do that, in Explora, the user first selects a pattern type, to indicate the interest in subgroups that deviate from some a priori or overall behaviour, or in subgroups that have changed their behaviour in the last 2 years, or in subgroups that show a special trend over the last 10 years, etc.. The next specifications relate to the search spaces that shall be constructed: subsets of data that are to be analysed are implicitely defined by selecting populations, several sets of variables and their values and taxonomies to construct ranges, target groups and subgroups. Further, search strategies and their parameters (e.g. exhaustive search of depth 3, beam search of depth 3 and width 5, etc.) are selected. Finally redundancy filters and quality functions can be selected when non default options shall be used. The potential for automatically set these specifications by the system in an intelligent way is low. Some possibilities exist for determining appropriate search strategies, their parameters, and quality functions, given the (size of the) search task and some additionally high level indications of the user. Such indications could refer to some properties of the findings like accuracy, homogeneity of subgroups, etc., and higher level goals could be transformed into the settings of some more technical parameters.

Also the domain knowledge that is exploited in Explora to deal with interestingness is very limited. This includes some data dictionary knowledge, especially the type of independent variables to preselect pattern types, and taxonomies which are mainly used to generate the results on an appropriate hierarchical level. Explora does not offer the option to define and exploit domain constraints, e.g. between values of variables that may (not) occur together or special forms of syntactical biases that go beyond some basic parameters of the description language (e.g. maximal and minimal numbers of conjunctions). Because Explora does not apply FOL description languages (QUINLAN [17]), the possibilities for domain constraints are limited anyway. Also, there are no features to deal with the novelty and usefulness aspect of interestingness in Explora, for instance by comparing current results with previously discovered results or studying the user behaviour when selecting or rejecting presented results.

In Explora, interestingness is evaluated locally in the search process by the verification method when processing a single pattern instance (node in the search space) and globally treating a set of instances by pruning criteria and search refinement (i.e. the suppression algorithm). An application test within the method verifies some constraints for interestingness. The suppression algorithm of Explora relates to another aspect of interestingness, namely *non-redundancy* treated by dissimilarity of findings (compare section 3). Refinement is restricted to the analysis of a search space belonging to a single pattern type. No options are offered to analyse and combine results belonging to search spaces of different pattern types.

A theory of evaluation functions that measure aspects of interestingness for hypotheses has been developed partially for the subgroup paradigm and the various statistical measures like strength, generality, coverage (of target elements). See (KLÖSGEN [11]) for details, where axioms and equivalences for evaluation functions are studied.

Besides the exploitation of domain knowledge for interestingness evaluation, a KDD system has to rely on the user who, in an explorative and interactive way, has to (re-) direct a discovery process based on intuitions stimulated by intermediate results. Explora provides only a limited functionality to operate on results and redirect succeeding search tasks. These tasks have to be defined "manually" by referring again to the various windows for specifying populations, range-, target- and independent variables.

Explora mainly assesses the interestingness of individual subgroups. To some limited degree, also the interestingness of a set of subgroups is evaluated, e.g. by the suppression algorithm and some criteria related to the degree of coverage and overlapping, significance and simplicity of a set of subgroups (compare Figure 3). Further approaches are needed to discover an optimal set of subgroups (or findings).

5. CONCLUSION: SOME PROBLEMS FOR KDD

The preceding sections showed that there is already a vast spectrum of data mining options in Explora that can be applied successfully for a broad range of applications. However, KDD is a still evolving research area, and some problems have to be solved to provide high quality discovery results. In this final subsection, we will present some of these problems.

Ensuring statistically valid discovery results is a goal that should be strictly observed in any KDD application, especially since a large scale search is performed. It must be excluded as far as possible that mere random results are generated by a KDD process. Traditionally, there are two areas for which statistics has already developed a framework for ensuring validity and providing some indication on the confidence the user can have in the results. For a classification task, classification accuracy of the whole set of derived findings (rules) is used as a single criterium to express and compare the quality of discovery results, and the variance of the accuracy is analysed to assess accuracy for applying classification rules for new data. Methods based on analysing separate training and test datasets (e.g. cross validation) dominate in this classification discovery task. When the discovery task consists in fitting data by equations, several parameters express the quality of a derived equation (e.g. variance reduction) and usually confidence intervals are derived for the parameters of the equation. Similar methods for assessing the validity and confidence of discovery results and for measuring the overall quality of e.g. a set of rules or subgroups are also necessary for descriptive patterns. As mentioned in Section 4, there are several criteria such as degree of coverage and overlapping, significance, simplicity, etc. that can be jointly used to assess the quality of a set of subgroups describing a target group. For descriptive patterns, it must be avoided further that the descriptions overfit the given data.

A second problem area relates to providing adequate description languages. We have already mentioned the limited expressive power of propositional, attributive languages. Sometimes first order based approaches can be helpful (Quinlan [17]). Constructive induction is another important approach related to this problem. Here, additional variables are constructed (dynamically during search) that are better suited to describe the given data. Especially for time and space related data, such derived variables can be useful when including descriptive terms based on means, slopes or other (time-) series indicators. Dynamically starting a separate search process for finding adequate terms of a description language is already successfully solved for discretization of numerical variables in some KDD systems.

Finding a best set of hypotheses among a large set of significant hypotheses is a third problem area. Methods such as tree approaches heuristically produce a set of rules arranged as a tree, but usually there may exist better rule sets.

We have already identified the integration of several aspects of interestingness as a problem area in Section 3. Future generations of systems will include discovered knowledge in their domain knowledge base to a still higher extent and use these findings for further discovery processes. They will incorporate more learning and adaptive behavior. Discovery methods will be used also to learn from the users by monitoring and analysing their reactions on the discovered and presented findings to assess the novelty facet of interestingness.

For very large datasets including millions of tuples and several hundreds of variables, data reduction is important. Although high performance solutions such as Data Surveyor (Holsheimer et al.[7]) can extend the applications of KDD methods from the usual boundaries of common KDD systems (10**6 tuples, a few hundreds of fields) by some orders, feature selection and sampling methods are often necessary to provide time efficient interactive discoveries. Interactivity of discovery systems is important because of the explorative nature of the KDD process (see Section 1). But also for ensuring clear discovery results, reduction of variables is necessary.

Explorative data analysis methods often underline the principle of robustness (compare Tukey and other protagonists). For KDD, this means that discovery results should not change too sensitively respective to small alterations of the data, description language or selected values of the dependent variables. As an example, consider the definition of the target group in Figure 3. If the specification `QKG1 < 0.16` would be slightly changed to `QKG1 < 0.17` (assuming no large distributional effects for the variable QKG1), this should not lead to a totally diverse set of rules identified in the discovery process. The main concern in KDD has been on accuracy, whereas robustness until now only plays a minor role in discovery research.

A variety of pattern types is available to execute discovery tasks such as identification of interesting subgroups. *Second order discovery* to compare and combine the results for different patterns could be necessary, especially if many analysis questions can be issued to the data. For example, when deriving classification and characteristic rules for a target group as separate discovery tasks, the discovery results should be harmonized, in the sense that the same variables are used for the different conditions.

A next point relates to changing data and domain knowledge. This problem area includes incremental mining methods adapting existing results according to the addition or modification of a small number of tuples and the comparison of new discovery results (for the new data) with the preceding results. The role of domain knowledge has been partially addressed in the preceding sections. One purpose of domain knowledge for KDD is restricting search to exclude uninteresting findings and to increase time efficiency. This technique has been mainly applied in ILP approaches, but should be more extensively applied also for the mainstream KDD methods.

Finally there are a lot of technical challenges to ensure efficient and interactive KDD processes. High performance solutions are necessary for VLDB applications. Other problems relate to the integration of KDD systems with other systems such as database systems or statistical packages. One promising development direction is characterized by incorporating KDD functionality within DBMS systems.

References

1. L. Breiman, J. Friedman, R. Olshen, C. Stone (1984). *Classification*

and Regression Trees. Belmont, CA: Wadsworth.

2. U. FAYYAD, G. PIATETSKY-SHAPIRO, P. SMYTH, and R. UTHURUSAMY (eds.) (1996). *Advances in Knowledge Discovery and Data Mining*, Cambridge, MA: MIT Press.
3. D. FISHER (1987). Knowledge Acquisition via Incremental Conceptual Clustering. *Machine Learning* **2**.
4. W. FRAWLEY, G. PIATETSKY-SHAPIRO, C. MATHEUS (1991). Knowledge Discovery in Databases: An Overview. *KnowledgeDiscovery in Databases*, EDS. G. PIATETSKY-SHAPIRO and W. FRAWLEY, Cambridge, MA: MIT Press.
5. F. GEBHARDT (1991). Choosing among Competing Generalizations. *Knowledge Acquisition* **3**.
6. F. GEBHARDT (1994). Interessantheit als Kriterium für die Bewertung von Ergebnissen. *Informatik Forschung und Entwicklung* **9**.
7. M. HOLSHEIMER, M. KERSTEN, A. SIEBES (1996). Data Surveyor: Searching the nuggets in parallel. *Advances in Knowledge Discovery and Data Mining*, EDS. U. FAYYAD, G. PIATETSKY-SHAPIRO, P. SMYTH, and R. UTHURUSAMY, Cambridge, MA: MIT Press.
8. W. KLÖSGEN (1992). Problems for Knowledge Discovery in Databases and their Treatment in the Statistics Interpreter Explora. *International Journal for Intelligent Systems* **7** (7).
9. W. KLÖSGEN (1993). *Explora: A support system for Discovery in Databases, Version 1.1, User Manual*, Sankt Augustin: GMD.
10. W. KLÖSGEN (1995). Efficient Discovery of Interesting Statements. *The Journal of Intelligent Information Systems* **4**, No 1.
11. W. KLÖSGEN (1996). Explora: A Multipattern and Multistrategy Discovery Assistant. *Advances in Knowledge Discovery and Data Mining*, EDS. U. FAYYAD, G. PIATETSKY-SHAPIRO, P. SMYTH, and R. UTHURUSAMY, Cambridge, MA: MIT Press.
12. W. KLÖSGEN and J. ZYTKOW (1996). Knowledge Discovery in Databases Terminology. *Advances in Knowledge Discovery and Data Mining*, EDS. U. FAYYAD, G. PIATETSKY-SHAPIRO, P. SMYTH, and R. UTHURUSAMY, Cambridge, MA: MIT Press.
13. J. MAJOR and J. MANGANO (1995). Selecting among rules induced from a hurricane database. *The Journal of Intelligent Information Systems* **4**, No 1.
14. R. MICHALSKI (1984). A Theory and Methodology of Inductive Learning. *Machine Learning*, EDS. R. MICHALSKI, J. CARBONELL, T. MITCHELL, New York: Springer-Verlag.
15. R. MICHALSKI and R. STEPP (1984). Learning from Observation: Conceptual Clustering. *Machine Learning*, EDS. R. MICHALSKI, J. CARBONELL, T. MITCHELL, New York: Springer-Verlag.
16. C. MATHEUS, G. PIATETSKY-SHAPIRO, D. MCNEILL (1996). Selecting and Reporting What is Interesting: The KEFIR Application to Healthcare Data. *Advances in Knowledge Discovery and Data Mining*, EDS. U.

Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Cambridge, MA: MIT Press.

17. R. Quinlan (1990). Learning Logical Definitions from Relations. *Machine Learning*, **5 3**.
18. G. Webb (1995). OPUS: An Efficient Admissible Algorithm for Unordered Search. *Journal of Artificial Intelligence Research (3).* Page 11