Discovery Planning: Multistrategy Learning in Data Mining

Kenneth A. Kaufman and Ryszard S. Michalski *

Machine Learning and Inference Laboratory
George Mason University
Fairfax, Virginia, 22030, USA
{kaufman, michalsk}@aic.gmu.edu

* Also Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland

Abstract

The process of applying machine learning to data mining may require many trials, backtracking, and multiple executions of different learning and inference procedures. Such a process can be time-consuming, laborious, and prone to errors. To overcome these problems, this paper proposes an integration of diverse learning and inference procedures into a system that can automatically pursue different data mining tasks according to a high-level plan developed by a user. The solution involves a meta-language, called KGL (*Knowledge Generation Language*), for specifying a data exploration process in terms of high-level operators and conditional statements that depend on the results of previous operators. Operators invoke corresponding machine learning and inference programs, and specify their parameters according to the current tasks and previous results. To assist in illustrating the outcomes of exploration they are organized into *association graphs (AGs)*, which can indicate logical, statistical and equational relationships in the data. The methodology is exemplified by preliminary results in the areas of demographics and medicine.

Introduction

A major challenge of machine learning research is the integration of a wide range of learning and inference strategies into a multistrategy system that is able to pursue different learning goals automatically. Depending on the goal and the available input information (which may come from a teacher, a database, or the environment), different sequences of learning and inference operators may be required to achieve that goal. This idea is particularly important for the task-oriented exploration and mining of large databases and data warehouses. In such applications, generation of the desired knowledge may require many trials, backtracking, and multiple executions of different learning and inference steps and strategies. This process can be time-consuming, laborious, and error prone.

The central problem in developing such a multistrategy learning/data mining system is how to integrate the different strategies in the pursuit of a given learning goal. There needs to be a control system able to decide which operators should be applied at any given step, and all operators must be implemented in such a way that the output from one can be an input to another.

Among the operators that may be called upon in data mining applications are operators to discover patterns that optimize a given *pattern quality criterion, determine exceptions* to patterns, generate a *characteristic description* or a *discriminant description* of a class of entities, determine the *most relevant attributes* for a given task, select the *most representative examples* from a very large example set, *conceptually cluster* cases into classes, generate a problem-oriented *decision structure*, *assign* class membership to a new instance, and automatically determine a *learning curve*.

In the case of data exploration and knowledge discovery, the issue is what kind of operators to apply to the data under which conditions, in which order, and using which parameters. One major challenge is to determine how to apply individual strategies and operators in a flexible and mutually dependent way, so that a desired solution can be obtained. Another challenge lies in determining which newly generated results should be regarded as useful.

The most attractive and long term solution would be to have a control system that given a data-mining goal specified abstractly by a user in natural language, could automatically synthesize and execute the most effective sequence of learning and inference operators for achieving that goal. In this research we have chosen a simpler approach in which the user defines the plan of data exploration in terms of high-level operator invocations. This plan is specified in a *knowledge generation language*, called KGL. The next section describes a system that provides a basis for implementing such a meta-language level integration of multiple learning and discovery operators.

INLEN

In the last few years, we have been developing the INLEN system for multistrategy data mining, knowledge discovery, and inference, based primarily on symbolic machine learning methods and techniques. INLEN integrates a range of *knowledge generation operators*, many of which represent different programs originally developed for use in stand-alone machine learning applications (Michalski et al. 1992; Michalski and Kaufman 1998). The use of these operators allows one to

discover general patterns, trends or exceptions in data that may not be apparent when only one type of strategy is applied. The results of applying diverse operators may also suggest a subsequent set of experiments that otherwise would not have been proposed.

While INLEN makes it easy for the user to apply diverse operators (corresponding to different machine learning and inference programs), the decision which operator to apply at any given step is made by a user. A disadvantage of such a user-level integration is that the data analyst needs to inspect the results of each step of the process in order to determine which operator to apply next. This process can be laborious and time-consuming, and prone to errors.

It is hypothesized that there exist general rules of control, which are applicable to a range of knowledge discovery problems. Such rules could be usefully be embedded within a learning and discovery system. One could also provide mechanisms for specifying domain- or task-dependent control rules. Further, individual users may have their own collections of repeating tasks to perform on the data. For these reasons, it is important to provide a means through which a user can articulate general plans for experiments to be conducted by a multistrategy data exploration system, beyond serial invocations of basic operators. This way the system could automatically perform complex sequences of actions in search of desirable knowledge.

Following this idea, we have developed KGL-1, a prototype meta-language for multistrategy data exploration. The language allows the user to create plans for guiding the system through various contingencies and executing different types of operators. The central motivation behind this language is to allow the user to write simple programs to accomplish very complex tasks. These programs can be executed once, periodically, on an infusion of new information into a database, or on the detection of some patterns in the data and/or knowledge.

The INLEN system integrates a relational database, a knowledge base, machine learning and statistical *knowledge generation operators*. To be able to perform easily operators that require both data and knowledge, the concept of a *knowledge segment* was developed (Kaufman, Michalski, and Kerschberg 1991). Knowledge segments are used for passing results from one operator to another and to the user. One research issue addressed in INLEN's development has been how to implement and utilize such knowledge segments. INLEN provides a user with the ability to search for and extract important and/or actionable knowledge from a database, to organize new knowledge from different viewpoints, to test it on a set of facts, and to facilitate its integration within the knowledge base. The tools for realizing these goals in INLEN are its knowledge generation operators (KGOs), which are designed to perform many types of learning and inference. They support the search for patterns, trends or exceptions in data.

While INLEN was the first such system, several recent data exploration systems have also adopted an integrated, multi-operator architecture, for example, KEPLER (Wrobel et al. 1996) and DBMiner (Han et al. 1996). The major differences between INLEN and them are INLEN's focus on a seamlessly integrated flow of knowledge segments, a wider range, and more powerful knowledge generation operators. Here is a summary of INLEN's operators:

- Ruleset generating operators—they take data and/or knowledge as an input, and return a ruleset characterizing the data. These operators differ from one another in the type and kind of ruleset generated (characteristic vs. discriminant; intersecting vs. disjoint vs. sequential).
- Decision tree generating operator—it takes a ruleset (edited within the system or generated by a ruleset generating operator), and returns a decision tree (or decision structure) optimized according to a criterion reflecting the decision-making situation (Michalski and Imam 1997).
- Classification generating operator—given a set of data, it creates a hierarchy of classes into which the data can be classified, and a set of rules characterizing the classes (it repeatedly applies a conceptual clustering method).
- Equation generating operator—it applies a method for qualitative and quantitative numerical discovery.
- Inferential operators—they perform selected inferential knowledge transformations, e.g., abstraction/concretion of numeric attributes, reasoning with structured attributes (Kaufman and Michalski 1996), ruleset optimization and matching rulesets with data.
- Representation space modifying operators—they change the given representation space by generating new attributes, abstracting attributes, and/or removal of less relevant attributes (constructive induction).
- Data modification operators—they change the given data set, e.g., by selecting a random sample of examples, or determining the most useful examples for a task.
- Ruleset performance evaluation operators—they determine the performance of a ruleset by applying the rules to testing data set and recording the results.
- Statistics generating operators—they determine selected statistical properties of data, e.g., a table of correlations among attributes.
- Knowledge visualization operators—they take data or rulesets and create a visual representation of them in a generalized logic diagram (a plane representation of a multi-dimensional discrete space).

Knowledge Generation Language (KGL)

Prior versions of INLEN made it easy to apply the available knowledge generation operators, but their application required explicit user decisions at every step. Most of these operators evoke machine learning programs that may require the specification of parameters to reflect accurately the nature and constraints of the problem at

hand. For complex and large databases, their execution may take considerable time and effort, and generate a large amount of output for analysis. Such a process also carries risks of errors and inefficiency.

To address these problems, we developed a meta-level language, KGL, for planning experiments in INLEN. In KGL, instructions can take the form, "If a certain ***condition*** is satisfied by a dataset or the current associated hypotheses, apply the following ***operators*** with the following ***parameters***." For example, if the database is continuously being updated, one may specify the following KGL instructions: "If more than 3% of the records in the database have been modified or added since the last application of this rule, test the knowledge base on the new data. If its degree of match falls below a 90% threshold, apply incremental learning to improve the knowledge base, and analyze the records that do not match the original rules to see if they share any common bonds."

KGL has been designed according to the following requirements:

- Every learning or knowledge processing program integrated into the system can be invoked by a single KGL operator.
- Properties of the database can be referred to in KGL statements (for example, "If there are 10% new examples of class A in the database, invoke a rule learning operator"; "Determine the percentage of missing values in the database.")
- Properties of the attributes, of the generated rules, the rulesets, and the decision trees can be referred to in KGL statements (for example, "Select nominal attributes with five or fewer legal values, and generate for them decision rules in terms of all numerical variables," or, "If the first rule in the generated ruleset covers over 95% of the training examples, remove all remaining rules from the ruleset.")
- Looping and branching are allowed as in conventional programming languages.

INLEN's knowledge base may contain the following types of knowledge: definitions of the variables and their types; domain constraints; generated rulesets and decision structures, data classifications, equations characterizing quantitative relationships; references to sets of examples in the database with an indication of their roles, e.g., training, testing, representative or exceptional; data and knowledge visualizations, and characterizations of the relationships among given concepts (an *association graph*—see below). INLEN's knowledge base has evolved a syntactic structure based on the VL1 variable-valued logic representation (Michalski 1973) and the ideas of Annotated Predicate Calculus (Michalski 1983). This structure can often represent complex relationships in a simple, easily interpreted way. Rulesets, rules and conditions are hierarchically arranged, and annotated by parameters and pointers to the data supporting or contradicting them (Kaufman and Michalski 1997).

Most previous efforts to create a high-level language for multistrategy learning and knowledge discovery have taken a Prolog-style logic programming approach. One notable exception is M-SQL (Imielinski, Virmani, and Abdulghani 1996), which extends the SQL data query language by adding the ability to query for certain types of rules and invoke a rule-generating operator. The KGL knowledge generation language differs from M-SQL in that it is able to call upon many different types of knowledge generation operators, and also in that it is designed to be less tightly coupled with SQL, and more closely resembles a programming language than a query language.

A language in which a different approach to the acquisition of knowledge is KQML (Finin et al. 1994). KQML is viewed as a tool by which intelligent agents may communicate among themselves and exchange the information needed to complete their individual tasks. It is thus designed to permit queries for individual pieces of knowledge. KGL differs from KQML in that it focuses more on queries for knowledge that fits a given abstract template (e.g., rules of a certain degree of strength), and that within the language one can call out to diverse learning and discovery operators to generate the knowledge base to be used.

A system similar to INLEN/KGL is CLEMENTINE, a data mining toolkit commercially developed by Integral Systems, Ltd. In CLEMENTINE, the user may specify a plan for a sequence of actions by a simple interface. KGL differs in that the language allows the specification of branching and looping conditions that may be based on the knowledge base or on locally assigned variables, and in employing different, more diverse, and in some cases significantly more powerful knowledge generation operators.

CLEMENTINE also supports knowledge visualization similar to the association graphs (AGs) described below. A major difference is that AGs allow the representation of multiple argument relationships, based on VL1 attributional language rules, using multiple links joined together when representing dependencies other than 1-to-1.

Also, the CLEMENTINE presentations are at a lower level of abstraction; nodes in the graphs represent individual attribute values. In the association graph methodology, nodes represent concepts or attributes as a whole, and the links are annotated to represent any monotonic tendencies between them.

Example of INLEN's Knowledge Structure

This example illustrates an application of INLEN to the analysis of the CIA World Factbook. The countries of the world were divided into classes based on their fertility rate, with the lowest class describing the 42 countries with fertility rates less than 2 per 1000 population, and other classes representing ranges of size 1, up to the highest class—countries with fertility rates above 7. One of the generated rules characterized 16 countries in the lowest fertility rate class, and differentiated them from the countries with higher fertility. This rule is shown in Figure 1.

Parameters *Pos* and *Neg* indicate the number of positive

and negative examples of the class satisfying a given condition. For example, "Birth Rate is between 10 and 20," characterizes all 42 low-fertility countries and 20 others. This representation schema leads to a natural division of a rule base into subunits, each described by a set of pertinent statistics (Ribeiro, Kaufman and Kerschberg 1995). An example of this structure is shown in Figure 2. The application of an operator (recorded as an individual learning *Session*) is associated with an input data set, the time at which it was run, and a set of output *Rulesets*. Each of these Rulesets is associated with the class of objects it describes, statistics on the training sets associated with it, and the individual *Rules* that make up the Ruleset. The Rules are associated with the training examples that satisfied them through *Rule Weights* and *Example Keys*, and also with the individual *Weighted Conditions* within the Rules. The knowledge segments corresponding to these Weighted Conditions can be associated with the *Condition* statement and the various pertinent *Condition Weights*, such as the number of positive and negative training examples that satisfy it. In this format, INLEN's knowledge base stores a set of decision rules.

Countries with Fertility ≤ 2 per 1000 population are characterized by: (42 examples)

	Pos	Neg
1. [Birth Rate is 10..20]	42	20
2. [Religion is R. Catholic or Orthodox or Anglican or Shinto]	24	31
3. [Infant Mortality Rate ≤ 40]	41	54
4. [Population Growth Rate ≤ 1 or 3..4]	32	56
5. [Literacy ≥ 70]	35	71
6. [Life Expectancy = 60..80]	41	92
7 [Death Rate = 5..15]	42	102
8. [Net Migration Rate ≥ -10]	42	140
Total:	*16*	*0*

Examples Covered: Andorra, Antigua, Austria, Belgium, …

Figure 1. A rule in INLEN's knowledge base, as presented to the user.

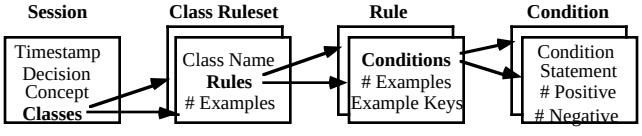


Figure 2. Knowledge organization for a decision ruleset.

Figure 3 shows an instantiation of this architecture - a knowledge segment from which the rule shown in Figure 1 was derived.

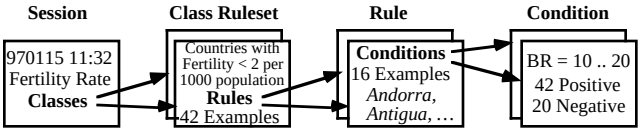


Figure 3. Exemplary instantiation of the schema presented in Figure 2.

The KGL interpreter is able to extract details from the knowledge base as requested, so that it may respond to queries for elements in the knowledge base such as:

- The conditions in rules for the class Fertility < 2 whose positive example coverage outnumber their negative example coverage by at least a 2 to 1 ratio.
- Rules for any concept (not only Fertility) that are satisfied both by Andorra and Antigua.
- Rules that include the condition Birth Rate = 10 .. 20.

An Example of KGL-based Multistrategy Data Mining

Preliminary versions of KGL have been applied to such datasets as collections of demographic characteristics of different countries and the results of surveys of parents and their children on various issues of growing up. In the demographic dataset, rules were first learned to characterize such concepts as low fertility rates (as seen above). Further KGL operators and commands performed such tasks as querying for and displaying strong rules (according to user-defined criteria); determining if a knowledge base meets certain quality standards, and improving it if it does not; and analyzing the makeup of a ruleset. A sample of KGL code for performing such tasks is shown in Figure 4, with output in Figure 5.

```
open PEOPLE
                {Select PEOPLE database}
do CHAR(decision=all, pfile=people1.lrn)
                {Characterize all concepts
                 using parameters specified in
                 file people1.lrn}
strongPGrules1 = #rules(PGR, %covd >= 60)
strongPGrules2 = #rules(PGR, num_covd >= 25)
strongPGrules3 = #rules(PGR,
     num_conds(supp >= 50 and pos > 10) > 2)
                {Count the strong PGR rules
                based on three metrics}
print "Number of strong PGR rules:
       Type 1 = ", strongPGrules1, ",
       Type 2 = ", strongPGrules2, ",
       Type 3 = ", strongPGrules3
if #conditions(Fert) > 150
                {Is Fert ruleset too complex?}
  begin
```

```
  do SELECT(attributes, decision=Fert,
       thresh=4, out=PEOPLE2, criterion=max)
              {find 4 best independent
               attributes}
  do CHAR(pfile=people1.lrn, decision=Fert)
              {then recharacterize}
  end
for i = 1 to 6
  begin
  print "Number of LE conditions with P/N
         ratio of at least", i, ":1 =",
         #conditions(LE, supp = 100 or
                     pos / neg >= i)
              {For each value of i from 1
               to 6, count and display number
               of conditions with a pos/neg
               coverage ratio of at least
               i:1.  The supp=100 condition
               avoids divide-by-0 trouble}
 end
```

Figure 4. Example KGL code for exploring a demographic database.

In the output in Figure 5, the number of strong rules, as determined by three criteria is displayed; the Fertility rule set is determined to be unsatisfactory, and is relearned using only the most relevant four independent attributes; and an analysis of the strengths of rules for Life Expectancy is presented.

```
Number of Strong PGR rules: Type 1 = 1,
Type 2 = 1, Type 3 = 7

Selecting best attributes from PEOPLE
for concept Fert -- Attributes chosen:
   Birth Rate
   Predominant Religion
   Life Expectancy
   Death Rate

Number of LE Conditions with P/N ratio
of at least 1:1 = 25
Number of LE Conditions with P/N ratio
of at least 2:1 = 10
Number of LE Conditions with P/N ratio
of at least 3:1 = 5
Number of LE Conditions with P/N ratio
of at least 4:1 = 1
Number of LE Conditions with P/N ratio
of at least 5:1 = 1
Number of LE Conditions with P/N ratio
of at least 6:1 = 1
```

Figure 5. Output from the Figure 4 KGL fragment.

From the survey dataset, we took 415 discrete-valued (either nominal or linearly ordered) attributes, and applied a KGL script that for each attribute in turn learned rules using them as decision attributes, after having reduced the dataset by invoking an operator to select the 25 attributes most likely relevant to the decision attribute, and then projecting the data accordingly. The program in Figure 6 handled this set of tasks.

```
begin
for i = 0 to 414
  begin
  open SURVEY
  do SELVAR(decision=i, out=SURVEY2,
            thresh=25, criterion=avg)
  do DISCSET(decision=0, scope=1, compile=no)
  end
end
```

Figure 6. A KGL program for survey data exploration.

Association Graphs

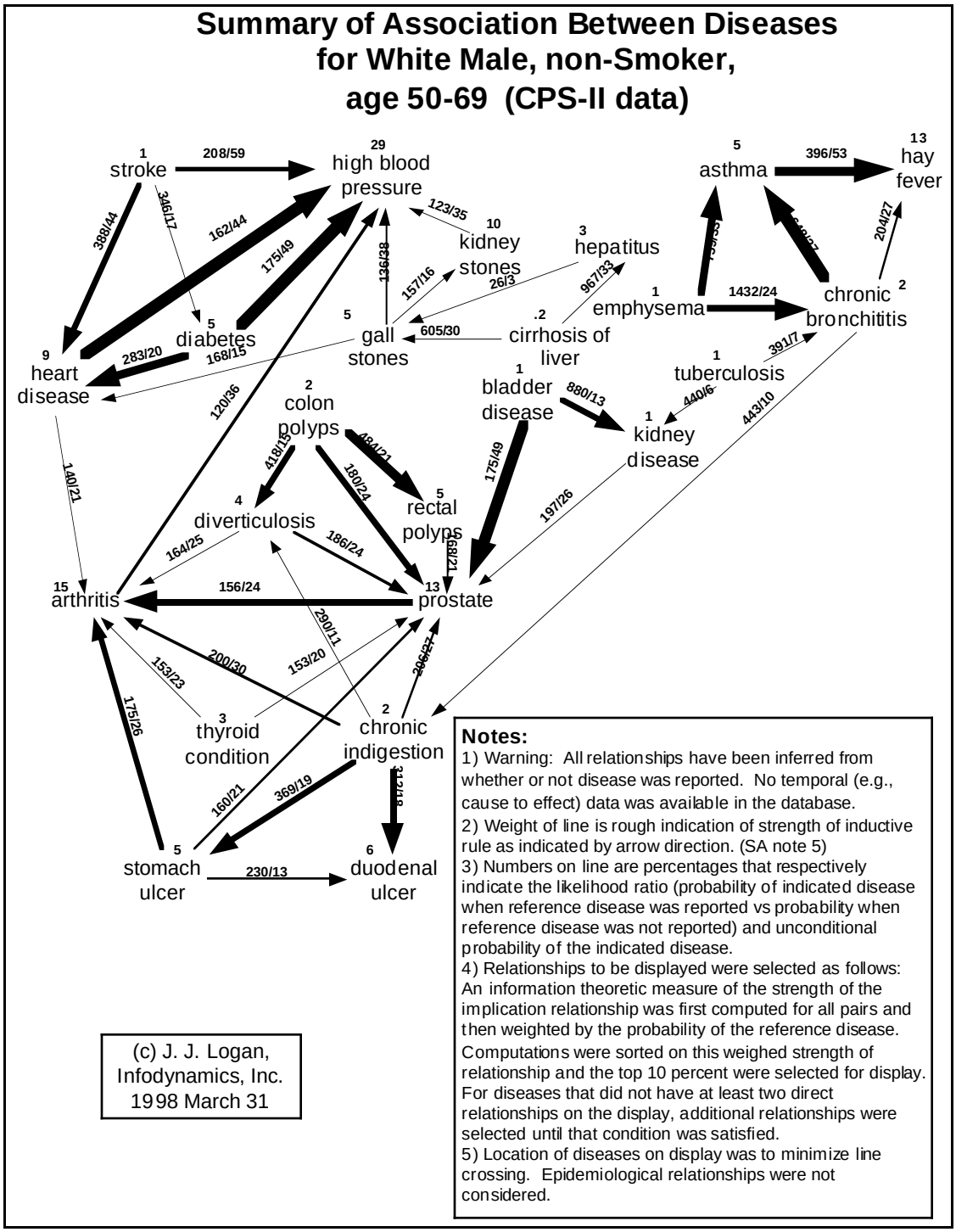
As mentioned above, exploring even a small database may result in large number of rules and associated information. For example, in the experiment with the survey data, rules totaling over 31 megabytes were generated from under 9 megabytes of data (in fixed-length field format). In order to easily grasp relationships among concepts, including both statistical and logical relationships, the idea of an *association graph* (AG) was proposed.

Nodes in such a graph may represent different concepts (e.g., individual attributes, inferred groups, or categories of attributes). Links in the graph represent different types of univariate or multivariate relationships at different levels of abstraction, depending on the type of association graph. An association graph may be categorized as statistical, logical or equational depending on the types of relationships represented by its links, or it may be a *general association graph*, combining different association types.

Links in an association graph may be directed to indicate the direction of stronger association. In the statistical AG shown in Figure 7, the links indicate the direction of higher conditional probability. In the logical AG shown in Figure 8, they indicate the direction from condition to action in a rule. The thicker the link, the stronger the relationship.

In an experiment whose goal is to find relationships among diseases and between diseases and lifestyles, the data source was a set of survey results collected by the American Cancer Society, with data on the occurrences of 25 diseases, as well as several background and lifestyle attributes. Initial experiments focused on the subset of the data consisting of surveys turned in by white male non-smokers, ages 50-65 (nearly 7500 records).

One study identified statistical relationships by generating weighted entropies between pairs of diseases, where the weighted entropy (WEEN) is defined as:



Graph developed by of J.J. Logan, Infodynamics, Inc., March, 1998. Reprinted with permission.

Figure 7. An example of a statistical association graph.

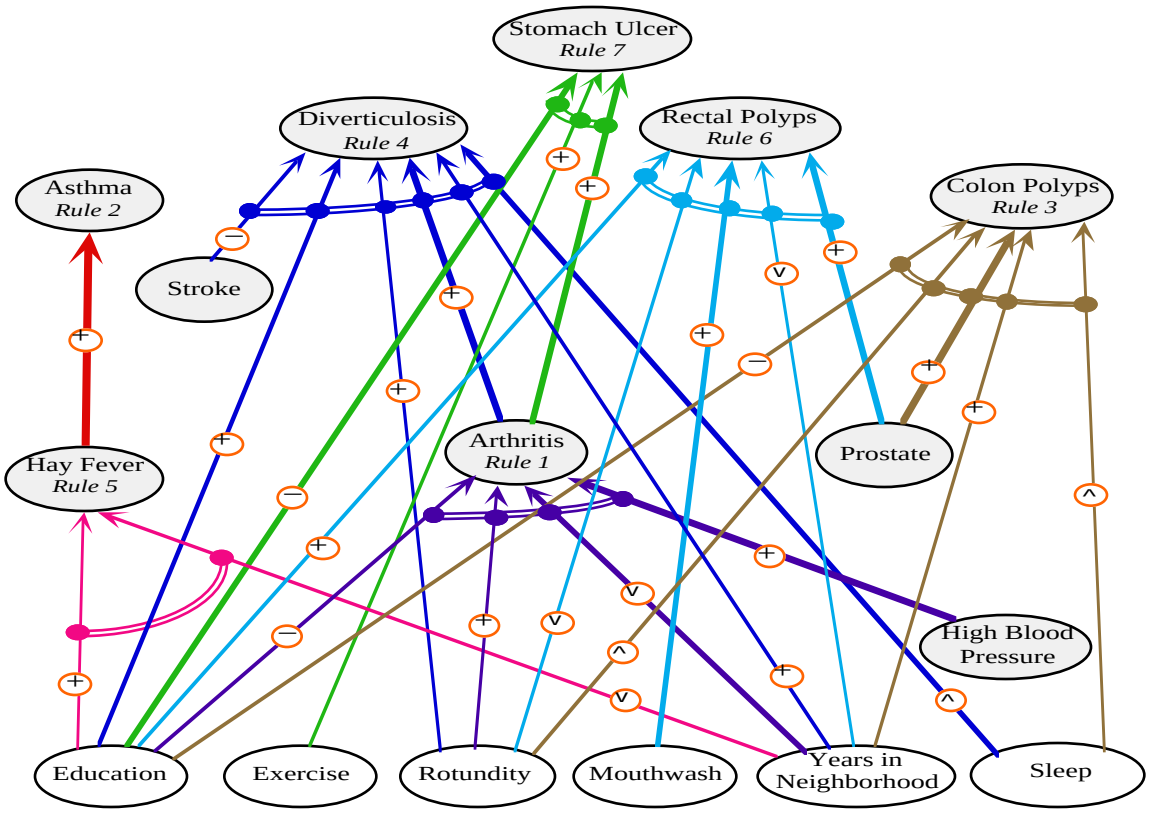


Figure 8. An example of a logical association graph.

$$WEEN(x,y) = p(x) * (p(y|x)*\log(p(y|x)/p(y)) + p(y'|x)*\log(p(y'|x)/p(y')))$$

An example of a univariate AG representing statistical relationships in a medical domain is presented in Figure 7. The links' thicknesses represent the magnitude of the weighted entropies, and the directions of the links indicate the higher conditional probability.

A second study used the AQ18 program (Michalski 1998) to generate decision rules in this domain. The strongest rules for seven of the diseases are represented in the association graph shown in Figure 8. In this logical AG, the thickness of the links represents the informativeness level of a condition in the rule (based on the ratio of positive training examples covered to total training examples covered). In addition, each link is annotated by a symbol that indicates the relationship between the condition and the action. A + or – suggests that an increase (decrease) in the attribute's value make the disease more likely, while a ^ or a v indicate that extreme values of the attribute make the disease more likely. Since most of the rules are multiconditional, arcs unite the conditions that form a common rule. This reiterates that a condition might not lead to a higher propensity for a disease except when combined with the other represented conditions.

In another experiment that used the results of parent-children surveys the source of the data was the National Youth Survey (United States), Wave I (Elliott 1977). This dataset catalogued the answers of parents and their children to various social and behavioral issues. It included socioeconomic data, records of problems at home or in the neighborhood, parental aspirations, attitudes toward deviance, discipline, community involvement, and drug use.

The goal of the INLEN experiment on this dataset was to determine which attributes (in this case, individual survey questions and their responses) were dependent on one another. A script was written that can be paraphrased as follows: For each decision attribute, log the attributes that appear in strong conditions of strong rules. Strong rules were defined as rules that covered more than 10 examples of the decision class, or over 40% of all of the training examples of that class. Strong conditions were defined as conditions in the rule whose support levels were over 40%, and were among the top three support levels of conditions in that rule. Support level is a measure of how well the condition alone is expected to imply the rule's conclusion, and is defined as the ratio of positive training examples of the decision class that satisfy the condition to the total number of training examples satisfying the condition.

In this case, a full association graph would have been too

complex with 415 attributes and a complex ruleset, so another step was taken to encapsulate the most densely populated portions of such a graph. This involved recording all maximal sets of attributes of size 3 or more, such that every attribute in the set was strongly dependent on every other attribute in the set by the criterion described above.

There were 284 such sets, of size 3 to 10. Six sets were of size 10, and each of them included the following attributes, indicating a knot of answers closely associated with each other:

YFBkRules (Does the youth think his family thinks he breaks rules?)
YFTrouble (Does the youth think his family thinks he gets into trouble)
YTBadKid (Does the youth think his teachers think he's a "bad kid"?)
YTBkRules (Does the youth think his teachers think he breaks rules?)
YTTrouble (Does the youth think his teachers think he gets into trouble)
YTBkLaw (Does the youth think his teachers think he breaks the law?)

There were other similar attributes (for example YFBadKid) that were less often grouped with these, or not at all.

The step of generating these groups of attributes exposed perhaps the most interesting piece of metaknowledge in this domain. Of the 415 attributes, about 60% were based on answers given by the child, about 30% on the answers given by the parent, and the rest on either observations by the interviewer or on geographic facts. We found that all of the 284 generated groups of 3 or more attributes were completely segregated with regard to source. In other words, if one attribute in the group was based on an answer provided by the parents, all attributes in the group would be based on the parents' responses. In the cases of tightly coupled attributes from different sources (e.g., parent's and child's ethnicity), no third attribute could be found that was strongly mutually associated with both of them.

Conclusion

The presented work describes a methodology for a language-level integration of a wide range of knowledge generation operators that employ machine learning and statistical programs. The proposed Knowledge Generation Language (KGL) supports the planning of complex learning and data mining experiments. Their results are visualized and summarized through association graphs (AGs).

KGL allows the user to write simple programs that can execute complex learning and knowledge processing tasks. It is modeled in part after common programming languages, and in part after database query languages such as SQL. A KGL user can access tools for such tasks as inductive learning from examples, feature selection, knowledge testing, and prediction of missing data based on provided or discovered knowledge. The language enables both the presentation of results to the user and the use of these results as a springboard to further discovery tasks. It allows the user easy interaction and communication with the program of the needs relating to the current application. It is not difficult to enhance the language by adding new operators or functions, and that is being done as new needs are recognized.

In its current form, KGL has several limitations. The presented version is a very new, and still unpolished solution, designed less for elegance than for proving its feasibility. It is planned that the KGL implementation will be refined and advanced in the future. The greatest strength of KGL is its ability to integrate and invoke diverse learning and knowledge processing operators and create a powerful environment for seeking out solutions to practical learning and discovery tasks. It is designed to serve as a novel and useful solution for implementing multistrategy learning capabilities. The experiments done so far are very promising, and indicate that the proposed methodology can be of very high practical utility.

Acknowledgments

The authors thank Jim Logan for the illustration presenting a statistical association graph. This research was conducted in the Machine Learning and Inference Laboratory at George Mason University, and supported in part by the National Science Foundation under Grant No. DMI-9496192.

References

Elliott, D. 1977. National Youth Survey (United States), Wave I, 1976 [Computer file]. ICPSR version, University of Colorado Behavioral Research Institute, Boulder, CO, 1977, distributed by Inter-University Consortium for Political and Social Research, Ann Arbor, MI, 1994.

Finin, T., Fritzson, R., McKay, D. and McEntire, R. 1994. KQML as an Agent Communication Language. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*. ACM Press.

Han, J., Fu, Y., Wang, W., Chiang, J., Gong, W., Koperski, K., Li, D., Lu, Y., Rajan, A., Stefanovic, N., Xia, B. and Zaiane, O.R. 1996. DBMiner: A System for Mining Knowledge in Large Relational Databases. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, 250-255.

Imielinski, T., Virmani, A. and Abdulghani, A. 1996. DataMine: Application Programming Interface and Query Language for Database Mining. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 256-261.

Kaufman, K. and Michalski, R.S. 1997. KGL: A Language

for Learning. *Reports of the Machine Learning and Inference Laboratory*, MLI 97-3, George Mason Univ.

Kaufman, K., Michalski, R.S. and Kerschberg, L. 1991. Mining for Knowledge in Data: Goals and General Description of the INLEN System. In Piatetsky-Shapiro, G. and Frawley, W.J. eds. *Knowledge Discovery in Databases*, 449-462. Menlo Park, CA: AAAI Press.

Michalski, R.S. 1973. Discovering Classification Rules Using Variable-Valued Logic System VL1. In *Proceedings of the Third International Joint Conference on Artificial Intelligence*, 162-172 Menlo Park, CA: International Joint Conferences on Artificial Intelligence, Inc.

Michalski, R.S. 1983. A Theory and Methodology of Inductive Learning. In Michalski, R.S. Carbonell, J.G. and Mitchell, T.M. eds. *Machine Learning: An Artificial Intelligence Approach*, 83-129. Palo Alto, CA: Tioga Publishing.

Michalski, R.S. 1998. The AQ18 Symbolic Learning and Data Mining Environment: Theory and Methodology. Reports of the Machine Learning and Inference Laboratory, George Mason Univ. (to appear).

Michalski, R.S. and Imam, I.F. 1997. On Learning Decision Structures. *Fundamenta Matematicae*, 31(1): 49-64.

Michalski, R.S. and Kaufman, K.A.. 1998. Data Mining and Knowledge Discovery: A review of Issues and a Multistrategy Approach. In Michalski, R.S., Bratko, I. and Kubat, M. eds. *Machine Learning and Data Mining: Methods and Applications*, 71-112. London: John Wiley & Sons.

Michalski, R.S., Kerschberg, L., Kaufman, K. and Ribeiro, J. 1992. Mining for Knowledge in Databases: The INLEN Architecture, Initial Implementation and First Results. *Journal of Intelligent Information Systems: Integrating AI and Database Technologies*, 1(1): 85-113.

Ribeiro, J., Kaufman, K. and Kerschberg, L. 1995. Knowledge Discovery from Multiple Databases. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, 240-245.

Wrobel, S., Wettschereck, D., Sommer, E. and Emde, W. 1996. Extensibility in Data Mining Systems. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, 214-219.