Natural Induction and Conceptual Clustering
A Review of Applications

Ryszard S. Michalski
Kenneth A. Kaufman
Jaroslaw Pietrzykowski
Janusz Wojtusiak
Scott Mitchell
Doug Seeman

MLI 06-3
P 06-3
June 12, 2006; Updated: August 23, 2006

NATURAL INDUCTION AND CONCEPTUAL CLUSTERING: A REVIEW OF APPLICATIONS

Ryszard S. Michalski*, Kenneth A. Kaufman, Jaroslaw Pietrzykowski, Janusz Wojtusiak,
Scott Mitchell, and Doug Seeman

Machine Learning and Inference Laboratory, George Mason University,
Fairfax, VA 22030-4444

*Also affiliated with Institute of Computer Science of the Polish Academy of
Sciences, Warsaw

{michalski, kaufman, jarek, jwojt}@mli.gmu.edu; samitchell@verizon.net; wdseeman@braemarnet.com
http://www.mli.gmu.edu

Abstract

Natural induction and conceptual clustering are two methodologies pioneered by the GMU Machine Learning and Inference Laboratory for discovering conceptual relationships in data, and presenting them in the forms easy for people to interpret and understand. The first methodology is for supervised learning (learning from examples) and the second for unsupervised learning (clustering). Examples of their application to a wide range of practical domains are presented, including bioinformatics, medicine, agriculture, volcanology, demographics, intrusion detection and computer user modeling, manufacturing, civil engineering, optimization of functions of very large number of variables (100-1000), design of complex engineering systems, tax fraud detection, and musicology. Most of the results were obtained by applying our recent natural induction program, AQ21, which is downloadable from *http://www.mli.gmu.edu/msoftware.html*. To give the Reader a quick insight into differences between natural induction implemented in AQ21 and some well-known learning methods, such as those implemented in C4.5, RIPPER, and CN2, as well as between conceptual clustering and conventional clustering, Sections 15 and 16 describe results from applying all these methods to very simple, designed problems.

Keywords: data mining and knowledge discovery, machine learning, natural induction, cluster analysis, conceptual clustering, data mining applications, machine learning applications

Acknowledgments

Research described here was conducted in the Machine Learning and Inference Laboratory at George Mason University, except for the example concerning tax fraud detection, which was done by our collaborator at Lockheed Martin Corporation, and the example of application to civil engineering, which was done by our collaborator at the Institute of Fundamental Technological Research at the Polish Academy of Sciences. Research and development of the ISHED system for optimizing heat exchangers was done by researchers of our laboratory, but its testing was conducted at the National Institute of Standards and Technology.

The Laboratory's research has been supported in part by the National Science Foundation under Grants No. IIS-9906858 and IIS-0097476, and in part by the UMBC/LUCITE #32 grant. In a few cases, presented results have been obtained under earlier funding from the National Science Foundation, the Office of Naval Research, or the Defense Advanced Research Projects Agency. The findings and opinions expressed here are those of the authors, and do not necessarily reflect those of the above sponsoring organizations.

1 INTRODUCTION

A common activity in almost all areas of science is to collect data in order to derive from them useful insights or discover new knowledge about the phenomenon under study. The amount of data collected may vary from very large, as in, for example, genomics, particle physics, or tax return, where it can be on the order of gigabytes or terabytes, to very small, as in archeology or criminology, where only a few, loosely linked facts may be available.

Modern tools for data analysis and data mining have evolved primarily from research in machine learning and statistics. The tools stemming from machine learning research are often available from university laboratories as free experimental computer programs. The tools stemming from research statistics, a much older discipline, are usually available as commercial, industrial-strength software, that was developed and is maintained by private companies.

Statistical and some machine learning tools for data analysis have been widely used, and are very useful in many practical applications. They have, however, significant limitations. Statistical tools do not work well with very small datasets and are primarily oriented toward creating quantitative (numerical) characterizations of the studied phenomena. These characterizations typically involve variables that are already present in the data, or some predefined functions of these variables. Both statistical tools and most of the current machine learning methods do not engage and reason with much prior domain knowledge while extracting patters from data. They also use relatively limited knowledge representation languages that may preclude them from discovering patterns or relationships in the data that a human expert employing a much richer language may be able to discover.

In many application domains it is desirable to characterize observations and express hypotheses about them not quantitatively, but rather qualitatively (descriptively), with an accompanying statistical annotation. Such qualitative descriptions are often quite sufficient for decision making, and may also be more reliable. For example, for everyday decision making, it is usually quite sufficient and even preferable to know that the next week will be sunny, warm, and with low humidity, rather than be given a set of numbers expressing precisely the temperature and humidity that is predicted for each day of the week. An accurate qualitative prediction is also usually much easier to create than an accurate quantitative prediction.

The GMU Machine Learning and Inference Laboratory is engaged in research, supported by the National Science Foundation, on developing a new, complementary approach to statistical data analysis, called *knowledge mining*, whose objective is to discover previously unknown regularities in data, and express them in the qualitative forms natural to people as they resemble those in which people express knowledge. These forms include logic-based or simplified natural language descriptions, and various knowledge visualizations, such as graphs, diagrams, figures, or images. Such forms are easy to understand, interpret, and use for creating mental models.

Basic methods of knowledge mining perform *qualitative data analysis* that seeks human-style explanations of data. They can efficiently derive task-relevant information from large volumes of data with many irrelevant facts, induce general rules and discover patterns in data, propose logical explanations of given facts, hypothesize structural relationships and causal dependencies, and incrementally improve the previously determined qualitative knowledge in the light of new data.

This article reviews examples of diverse practical applications of two methodologies of knowledge mining developed in our laboratory, natural induction and conceptual clustering. The presented examples include applications to bioinformatics, medicine, agriculture, volcanology, demographics, intrusion detection and computer user modeling, manufacturing, civil engineering, optimization of functions of very large number of variables (100-1000), design of complex engineering systems, tax fraud detection, and musicology. The article is written in a simple, tutorial style in order to make it easy to understand by a wide range of readers. Technical details and algorithms are described in papers listed in the References. Most of the referred papers are downloadable from www.mli.gmu.edu (select "Papers"). The learning program AQ21 that was used in many experiments is also downloadable from that website (select "Software").

Sections 2-11, describe selected applications of natural induction, Sections 12-14 describe applications of conceptual clustering, and Sections 15-16 present simple examples of comparative studies of natural induction and other methods. Most of the applications represent very recent work.

Because the concepts of "natural induction" and "conceptual clustering" are relatively new, and may not be familiar to the reader, we start by briefly explaining them. Natural induction is a form of supervised learning, a.k.a learning from examples, that hypothesizes general concept descriptions from concept examples or discovers patterns in data, in qualitative forms that are easy for people to understand, interpret, and make a mental model of them. Specifically, knowledge derived from data is expressed in terms of *logic-style rules* that directly correspond to simple natural language statements and are visualized using new forms of graphical representation, such as *concept association graphs*, *generalized logic diagrams* and *ruletrees* (see examples in Figures 3, 4, 6, and 7).

An important aspect of natural induction is that it places equal emphasis on predictive accuracy and on the understandability of computer-generated knowledge, in contrast to most of machine learning and data mining methods in which high predictive accuracy is the main, or only objective. Natural induction thus aims at being a "transparent box" method for analyzing data, in contrast to "black box" methods that may produce accurate results, but are opaque and difficult to interpret.

Our newest program implementing natural induction is AQ21 (Wojtusiak, 2004; Wojtusiak et al., 2006). Given a set of data, it outputs hypotheses learned from the data in the form of *rulesets* in *attributional calculus*, a logic and representation system developed for supporting natural induction (Michalski, 2004). For an example of such a ruleset see Figure 2.

Depending on the setting of its parameters, AQ21 may generate different types of data characterizations, such as complete and consistent generalizations, patterns that represent strong regularities, but partially inconsistent with the data, descriptions with exception clauses, and some other. An AQ21's learning process can be viewed as a search for a description (a ruleset) that maximizes a given measure of *description utility,* defined by a *Lexicographic Evaluation Functional*[1] (briefly, LEF):

$$\text{LEF} = <\text{QUALITY}, q\%; \text{SIMPLICITY}>$$

where:

QUALITY measures a description quality, Q(w), defined as:

$$Q(w) = \text{Cov}^{w} * \text{Config}^{1-w}$$

In this measure:

$\text{Cov} = p/P$ (“Coverage”)

$\text{Config} = ((p / (p + n)) – (P /(P + N))) * ((P +N) /N)$ (“Confidence gain”)

p and n are numbers of positive and negative examples in the training set covered by the description

P and N are total numbers of positive and negative examples in the training data, respectively, and

w is a parameter that allows the user to control the relative importance of the Cov and Config components.

SIMPLICITY is the reciprocal of the description COMPLEXITY, defined as the sum of the costs associated with each operator in the description. In our experiments, the default costs of operators were: conjunction – 4, disjunction – 10, internal disjunction – 2, range – 2, less than or greater than operators – 1, equality operator – 1, inequality operator – 2. For operators within an exception clause, the costs are doubled. When a ruleset consists of *m* rules, it is assumed that it has *m*-1 disjunction operators.

Parameter q%, called the tolerance for QUALITY, defines the range of QUALITY values of rules that will be evaluated for SIMPLICITY. Rules whose QUALITY is not within q% of the best rule in the set of candidates are ignored. LEF thus provides a simple multi-criterion evaluation of a set of alternatives. It is particularly attractive when there are many alternatives to evaluate, because the number of alternatives to consider decreases after applying each criterion. Such a situation often arises in the inductive inference from non-trivial data.

Conceptual clustering is a form of unsupervised learning (a.k.a. learning from observation) that concerns grouping observed entities into “conceptual clusters” that represent simple concepts, in contrast to conventional clustering, which clusters objects into groups of similar objects, according to some an a priori defined mathematical measure of similarity. Conceptual clustering outputs both clusters and cluster descriptions, and evaluates clusters on the basis of the quality of these descriptions, whereas conventional clustering outputs only clusters, and evaluates them on the basis of the intra- and inter-cluster similarities.

[1] For readers unacquainted with the concept of LEF (Michalski, 1972), here is a brief explanation. LEF can be used for ranking individual rules or entire rulesets. Let us assume that it is applied to ranking a set of rules. First, LEF determines rules that score the highest on the first criterion (in the above example, QUALITY). Rules whose QUALITY is at least (TopQuality - q% x TopQuality) are then evaluated on the second criterion (here, SIMPLICITY); others are rejected. The rule that scores the highest on SIMPLICITY is selected as the overall best according to the LEF. LEF can be applied to rank rules or any other entities based on multiple criteria.

Conceptual clustering is accomplished by executing a search for a clustering (collection of clusters) that optimizes a criterion of clustering quality that reflects clusters' "conceptual cohesiveness" (Michalski and Stepp, 1983a,b; Seeman and Michalski, 2006). Examples of the application of conceptual clustering are presented in Sections 11-13.

In our research on natural induction and conceptual clustering, the underlying description language is *attributional calculus,* which combines elements of propositional logic, predicate logic and many-valued logics for the purpose of facilitating inductive learning and qualitative data analysis (Michalski, 2004). The relationships discovered by natural induction may combine descriptive and statistical information. Here is an example of a hypothesis produced by the AQ21 learning program that analyzed of a medical dataset representing a gene microarray of patients with medulloblastoma:

> *If in a gene array derived from a patient, the expression of Gene-1611 is below the threshold T1, the expression of Gene-1036 is in the range T2 to T3, and the expression of Gene-914 is below the threshold T4,* ***or*** *the expression of Gene-1783 is above the threshold T5, then the patient's cancer is likely to be metastatic.*

For a detailed explanation of this result, see the next section. The above text is a direct translation of attributional rules learned by the program.

In the following sections, we describe examples of applications of our methods of natural induction and conceptual clustering to bioinformatics, medicine, agriculture, volcanology, demographics, intrusion detection and computer user modeling, manufacturing, engineering, musicology, and tax fraud detection. For readers interested in getting a quick insight into differences between our methods and some other well-known methods, Sections 14 and 15 use simple designed problems to compare the methods. Section 14 compares natural induction to other methods of supervised learning, and Section 15 compares conceptual clustering to a similarity-based method of unsupervised learning.

2 AN EXAMPLE OF APPLICATION TO BIOINFORMATICS

This example concerns an application of natural induction to the problem of diagnosing medulloblastoma from patients' gene microarrays (representing degrees of expressions of patients' genes). Medulloblastoma is a highly invasive primitive neuroectodermal tumor of the cerebellum and the most common malignant brain tumor of childhood. The data for this application were obtained from the Gene Expression Omnibus (GEO), NCBI NLM NIH, available online at http://www.ncbi.nlm.nih.gov/geo. The original gene microarray data consists of 46 records split into two classes: 20 records representing patients with metastatic tumors and 26 records representing patients with non-metastatic tumors. Each record registers values of 2059 real-valued attributes that represent the expression of different genes.

In the experiments that inspired this work, performed by McDonald et al. (2001), out of 2059 genes (serving as attributes), the authors selected the 87 with the highest *Prediction Strength Correlation,* defined as the ratio of the difference between the mean values in the two classes (metastatic and non-metastatic) to the sum of the standard deviations in the classes. Figure 1 shows a subset of a gene microarray with medulloblastoma data for a subset of 23 patients. Each row corresponds to a patient, and registers expressions of the selected 87 genes, represented in the columns.

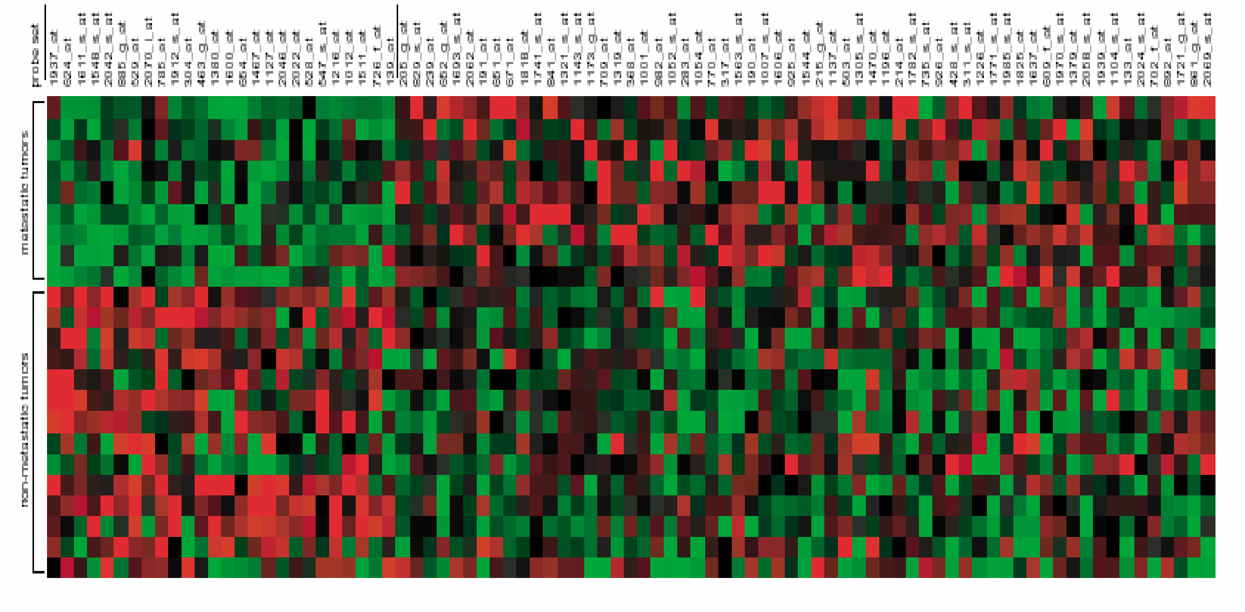


Figure 1: A gene microarray containing medulloblastoma data for 23 patients.

The first 9 rows represent patients with metastatic tumors, and the remaining 14 with non-metastatic tumors. Bright red color spots represent a high gene expression level, and bright green spots denote a low expression level.

For our experiments, we selected only the ten attributes that scored highest on the PROMISE measure of attribute quality, studied by our former student Baim (1982). The measure expresses a degree to which an attribute differentiates between classes. After projecting data on the selected 10 attributes, we applied the AQ21 natural induction program to hypothesize rules for distinguishing between metastatic and non-metastatic rumors.

In the experiment reported here, the training data for metastatic tumors consisted of 16 unique examples, and for non-metastatic tumors 12 unique examples (after attribute selection, some examples became indistinguishable). Given these examples, AQ21 discovered two rules (a ruleset) for the metastatic tumor presented in Figure 2.

```
[Cancer = metastatic]
    <=  [Gene-1611 <= 100.9: 18, 8, 69%, 18, 8, 69%] &
        [Gene-1036  = -41.76..160.8: 18, 20, 47%, 16, 4, 80%] &
        [Gene-914   <= 121.5: 20, 15, 57%, 16, 0, 100%]:
        #positives = 16, #negatives =0, #unique = 14, QUALITY = 1, COMPLEXITY = 17

    <=  [Gene-1783  >= 96.6: 6,0,100%,6,0,100%]:
        #positives = 6, #negatives= 0, #unique = 4, QUALITY = 1; COMPLEXITY = 5
```

Figure 2: An example of a ruleset discovered by AQ21 for recognizing metastatic tumors.

In Figure 2, the condition [Cancer = metastatic] is the rule's *consequent* that is implied by two alternative premises (that follow the implication sign "<="). A pair, a consequent and a premise, constitutes a single rule. Thus, Figure 2 presents two rules. The premise of the first rule is a logical conjunction of three conditions, and the premise of the second rule consists of just one condition.

The first rule states that if the expression of the gene Gene-1611 (interferon IFN-γ) is equal to or below 100.9, ***and*** the expression of the gene Gene-1036 (IL15: interleukin 15) is between -41.76 and 160.8 (inclusive), ***and*** the expression of the gene Gene-914 (ERG: v-etserythroblastosis virus E26 oncogene like, avian) is equal to or below 121.5 in a gene array of a patient, then metastatic cancer is indicated in that patient. The second rule states an alternative condition indicating medulloblastoma, namely, when the expression of Gene-1783 (RIN2: Ras and Rab interactor 2) is above or equal to 96.6.

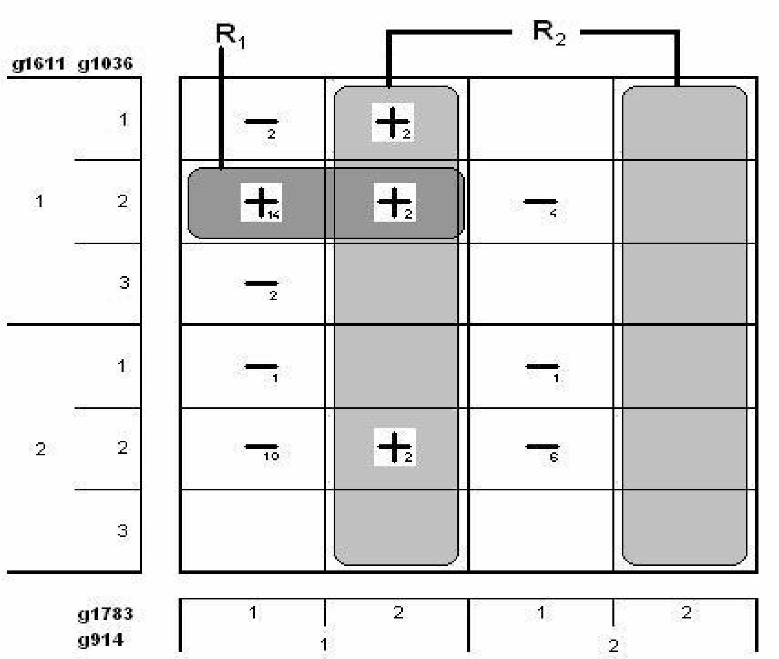
Each condition of every rule is annotated by two triples of numbers, listed after the colon. The first number in the first triple indicates the number of positive examples in the training set (here, the number of metastatic patients) that satisfy this condition (denoted generally as "p_c"), the second number indicates the number of negative examples (here, non-metastatic patients that satisfy this condition (denoted generally as "n_c"), and the third number is the condition *confidence*, defined as p_c /($p_c + n_c$), and expressed as a percentage.

The numbers in the second triple represent the same quantities but not just for the given condition, but for the logical conjunction of the given condition and all the previous conditions in the rule's premise. Thus, in the first condition of each rule, both triples are always identical, because there are no previous conditions because it is the first condition. However, the subsequent conditions in the premise will usually contain different triples, because they refer now to the numbers of cases (here, patients) that satisfy both the given condition and all the previous ones.

The five numbers at the end of each rule denote respectively: the total number of positive examples (here, patients with metastatic tumors) covered by the rule (#positives or, briefly, p), the total number of negative examples covered (#negatives or, briefly, n), the number of positive examples covered only by this rule and no other rule (#unique, of briefly, u), and the rule QUALITY and COMPLEXITY, as defined before. Note that when p and u parameters of a rule are very small, this indicates that the example covered by this rule is an outlier and may be an error. If n=0, the rule is fully consistent with all the training data.

To illustrate graphically rulesets generated by AQ21, we have developed two programs, KV ("*Knowledge Visualizer*"), which represents rules in a *Generalized Logic Diagram* (GLD), and CAG (*Concept Association Graph*), which represents them as a labeled graph with varying thicknesses of the links.

A GLD is a planar representation of a multi-dimensional space spanned over discrete attributes. For example, Figure 3 presents a GLD for an instance space spanned over three attributes, representing discretized values of expressions of genes, g1611, g1036, g1783, and g914 (the domains of these attributes have been discretized into 2, 3, 2, and 2 ranges, respectively).



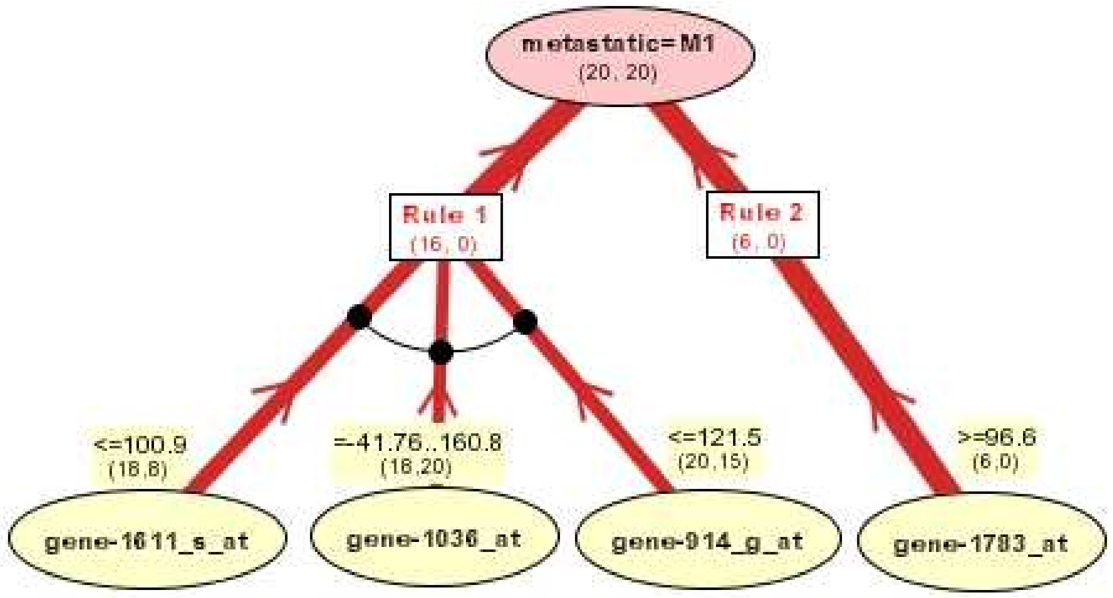
Cells marked by "+" and "-" represent examples of metastatic and non-metastatic cancers, respectively.

Figure 3: A general logic diagram visualizing training data and discovered rules for diagnosing metastatic tumors.

Each cell of the diagram represents a combination of values of these four attributes. Cells marked by a "+" represent metastatic patients, and those marked by a "-" represent non-metastatic patients. A small number next to a symbol "+" or "-" in a cell denotes the number of patients with metastatic and non-metastatic patients, respectively, whose discretized gene expressions are represented by this cell. For example, the cell defined by: g1611 = 1 & g1036 = 2 & g1783 = 1 & g914 = 1 represents 14 metastatic patients. In Figure 3, the first rule from Figure 2 is visualized as R1, and the second as R2. As one can see, a GLD displays both examples and rules in a simple, easy to understand form. This visualization method is, however, limited to cases in which the variables spanning the diagram are discrete and their number is relatively small.

To visualize more complex cases, we developed an alternative method, which employs a *concept association graph*. In such a graph, nodes represent attributes or attribute-value pairs, links represent rule conditions, and collections of links connected by an arc represent attributional rules. The top node represents a ruleset for the output attribute value indicated in the node. The thicknesses of the links represent a requested measure of the condition or rule importance. Depending on the parameter setting, the program can use different importance measures, such as *confidence* (*p /(p + n)*) or *support* (*p*), where p and n are the numbers of positive and negative examples, respectively, covered by the condition or rule.

For example, Figure 4 presents a concept association graph visualizing the discovered ruleset for medulloblastoma. The lowest nodes represent input attributes, and annotations on the links connecting them to rectangular nodes marked "Rule 1" and "Rule 2" represent conditions on the individual attributes.



The thickness of links is proportional to condition or rule confidence (p/(p+n)).

Figure 4: A CAG visualizing discovered rules for recognizing metastatic tumors.

The thickness of each link in this graph is proportional to the confidence of the corresponding condition or rule, thus this is not an ordinary labeled graph, but a graph that conveys information also through the thickness of the links and the arches connecting the links. The pairs of numbers (p,n) associated with the links, or inside the nodes denoted "Rule 1" and "Rule 2", indicate the number of positive and negative examples, respectively, covered by the corresponding condition or rule. For example, the pair (16,0) inside of the node marked "Rule 1" and the pair (6,0) inside of the node marked "Rule 2" indicate that these rules cover 16 positive and 0 negative examples, and 6 positive and 0 negative examples, respectively. Both rules have thus confidence p/(p+n) =1, and the links to the top node (metastatic =M1) are maximally thick.

<u>Comments on the results</u>

Let us briefly comment on the example of the hypothesis (ruleset) in Figure 2, discovered by AQ21 for recognizing metastatic patients in a gene array. The ruleset involves only 4 genes out of 2059 genes. When the experiment was performed using 5-fold cross-validation[2], the predictive accuracy was about 95%. (There was only one misdiagnosis. Because there was one example in the data that looked like an outlier with respect to other examples of the same class of patients; therefore, one cannot exclude the possibility that it might have been an example erroneously classified in the data).

---

[2] In 5-fold validation, the complete dataset for each class is split into 5 roughly equal groups. An experiment is repeated for each set of 4 groups serving as a training set, and the remaining 1 group serving as the testing set. The output predicative accuracy is the average of the predictive accuracies from each combination of the training and testing sets.

The two-rule ruleset is surprisingly simple and easy to understand, Despite having a very limited number of training examples to learn from and large number of attributes spanning the original search space, the ruleset discovered by the natural induction method achieved high predictive accuracy (95%).

These results from natural induction are in contrast with the neural net developed for the same task, which was described in (MacDonald et al., 2001). Their neural net requires measuring 80 genes (attributes), rather than 4, and its reported predictive accuracy is about 72%, lower by 23% than that of AQ21 rules. In addition, the neural net is a “black box” solution that is difficult to interpret and understand, while the “transparent box” ruleset obtained by natural induction can be easily interpreted, and executed even without a computer. Further research is needed, however, to test the rules obtained by natural induction on more medical data, and to have them evaluated by medulloblastoma experts before they could be accepted as a valid medical discovery.

3 AN EXAMPLE OF APPLICATION TO MEDICINE

This work applied natural induction to a problem of determining relationships between lifestyles and diseases of non-smoking males, aged 50-65. The study employed a database from the American Cancer Society that contained 73,553 records of responses of patients to questions regarding their lifestyles and diseases. Each patient was described in terms of 32 attributes: 7 lifestyle attributes (2 Boolean, 2 numeric, and 3 rank), and 25 Boolean attributes representing diseases.

The natural induction program AQ19 (a predecessor of AQ21) was applied to discover patterns (that is, tendencies, rather than unbreakable rules) characterizing the relationships between the lifestyles and 25 diseases, and possible relationships between the diseases. Among the many discovered patterns, a typical example is shown in Figure 5.

```
[Arthritis = Present]
   <=        [HBP=present: 432, 1765] &
             [Rotundity>=low: 1070, 5578] &
             [Education<=college_grad: 940, 4529] &
             [YinN > 0: 1109, 5910]: p = 325, n = 1156; P = 1171, N = 6240
```

where

HBP stands for High Blood Pressure

Rotundity is a discretized ratio of the patient’s weight to his height

YinN_denotes the years the patient lived in the same neighborhood

The two numbers listed within each condition after the colon denote p_c and n_c, that is, the number of positive and negative examples in the training set covered by that condition, respectively

p and n, are the number of positive and negative examples in the training set covered by the rule, respectively.

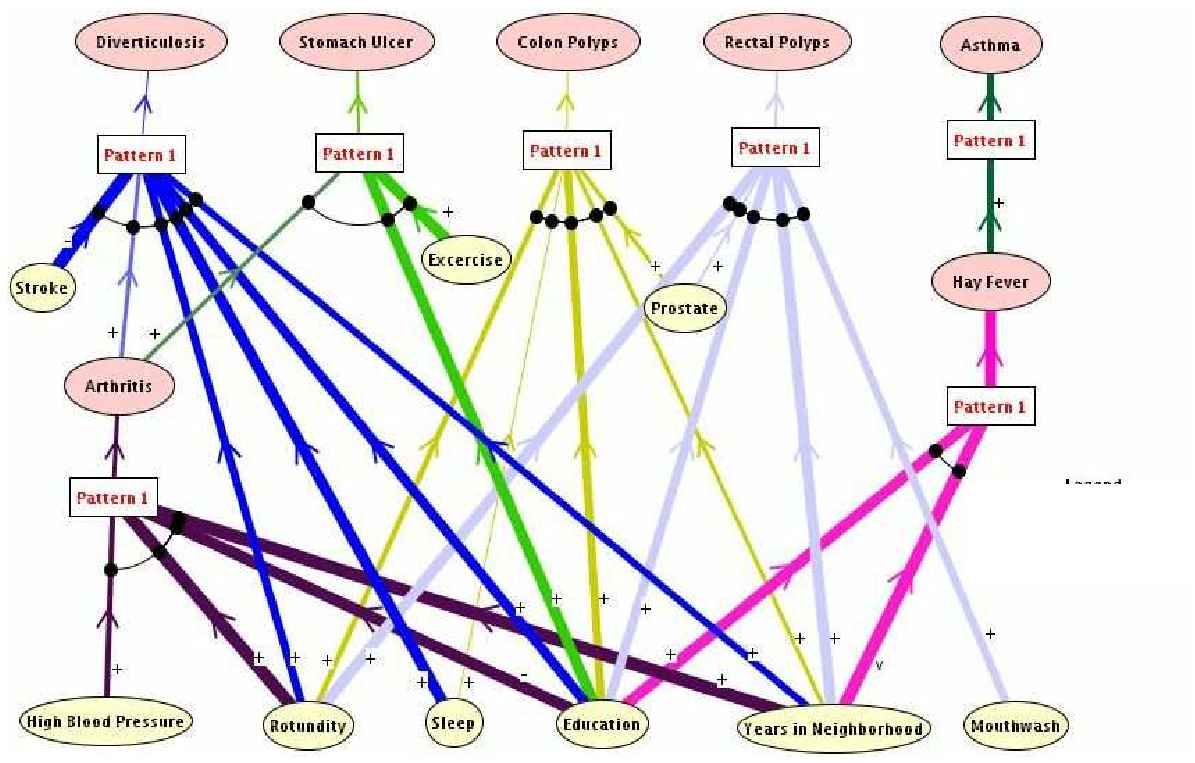
P and N are the number of positive and negative examples in the training data for that class (here, Arthritis), respectively.

NOTE: This rule was obtained by an earlier version of the program that did not output all the annotation parameters that were discussed in the medulloblastoma application.

Figure 5: A pattern for Arthritis discovered in the medical database.

The pattern in Figure 5 defines a set of conditions under which patients had arthritis relatively frequently, which include the presence of high blood pressure, higher than low "rotundity", no education beyond college, and staying in current neighborhood at least one year. In the training data, about 16% of the patients had arthritis (P/(P+N)), but among patients satisfying the pattern, the percentage grows to 22% (p/(p+n), that is, the likelihood of them having arthritis increases by 37%. The most significant factor in this pattern is high blood pressure, which by itself has confidence of about 20% (p_c/p_c+n_c).

Discovered patterns were visualized using concept association graphs. One such graph is presented in Figure 6. It was automatically generated from the discovered patterns using computer program, CAG, designed for graphically representing attributional rules. In this graph, a link's thickness reflects the condition coverage (p), and the link's annotation (+, –, v, or ^) indicates the type of the relationship between condition and consequent. Specifically, "+" represents a positive monotonic relationship (higher values of the condition attribute indicate higher values of the consequent attribute),"-" represents a negative relationship (lower values of the condition attribute indicate higher values of the consequent attribute), and "v" and "^" indicate that extreme values of the attribute indicate higher or lower values of the consequent attribute, respectively.



The thickness of links is proportional to condition or rule coverage.

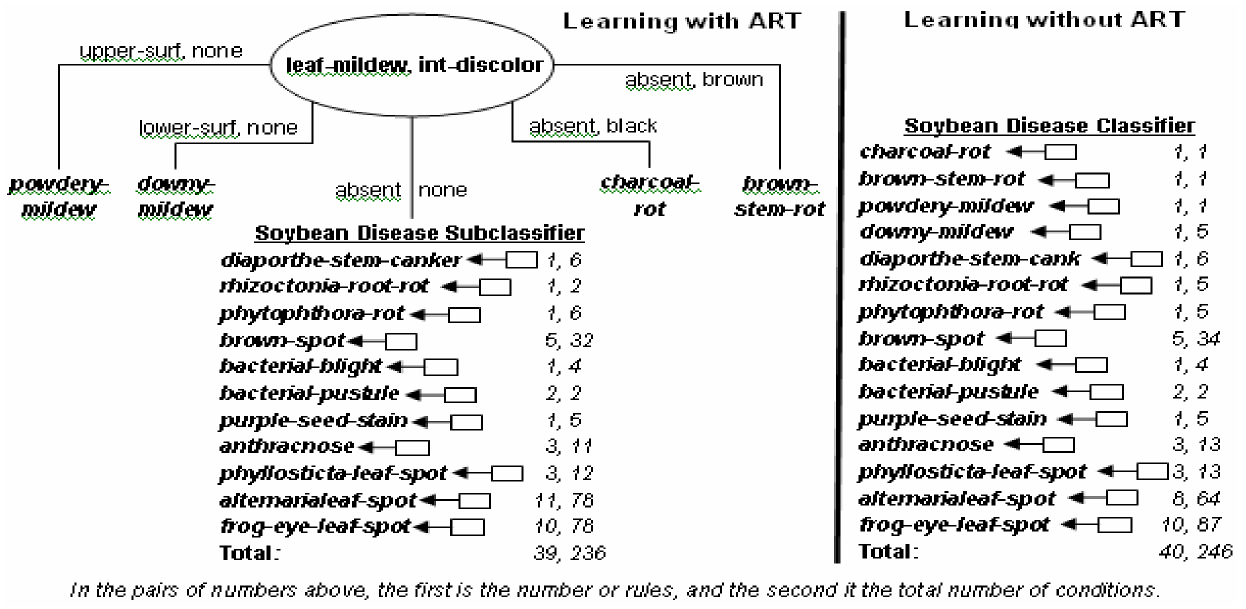
Figure 6: Concept Association Graph representing seven patterns in the medical database.

While no claim is made as to the practical usefulness of these specific obtained results, they indicate, however, that the developed methodology is potentially capable of discovering important patterns in the data and representing them in an understandable way, either as qualitative relationships or in graphical forms.

4 AN EXAMPLE OF APPLICATION TO AGRICULTURE

This section illustrates a form of natural induction that generates *attributional rule-tree*, rather then ruleset representations of concepts. An attributional rule-tree is a combination of a tree structure and a ruleset structure. It was developed as a simple representation of classifiers that classify entities into many related classes. It is intended for situations in which a flat ruleset or a decision tree might not be easy to mentally follow, but a rule-tree might represent related concepts in a more understandable way.

In this application, the task was to learn a classifier for diagnosing the most common soybean diseases (15 diseases) from a database describing disease cases in terms of 35 multi-valued attributes. The training data consisted of 266 cases provided by a domain expert. Figure 7 presents a rule-tree and an equivalent ruleset learned for the fifteen soybean diseases.



Soybean Disease Subclassifier

diaporthe-stem-canker	1, 6
rhizoctonia-root-rot	1, 2
phytophthora-rot	1, 6
brown-spot	5, 32
bacterial-blight	1, 4
bacterial-pustule	2, 2
purple-seed-stain	1, 5
anthracnose	3, 11
phyllosticta-leaf-spot	3, 12
alternarialeaf-spot	11, 78
frog-eye-leaf-spot	10, 78
Total:	39, 236

Soybean Disease Classifier

charcoal-rot	1, 1
brown-stem-rot	1, 1
powdery-mildew	1, 1
downy-mildew	1, 5
diaporthe-stem-cank	1, 6
rhizoctonia-root-rot	1, 5
phytophthora-rot	1, 5
brown-spot	5, 34
bacterial-blight	1, 4
bacterial-pustule	2, 2
purple-seed-stain	1, 5
anthracnose	3, 13
phyllosticta-leaf-spot	3, 13
alternarialeaf-spot	8, 64
frog-eye-leaf-spot	10, 87
Total:	40, 246

In the pairs of numbers above, the first is the number or rules, and the second it the total number of conditions.

Figure 7: Complexity comparison of rule trees and rulesets in the soybean domain.

The classifier in the form of a rule-tree was learned by the ART program based on the method introduced in (Michalski, 2002). ART works in two steps: the first step seeks *partitioning attributes* whose combination of values split given decision classes into different groups, and the second step applies an AQ learning program like AQ21 to attributional learn rules distinguishing classes within groups obtained in the first step.

For the soybean disease diagnosis problem, ART found two partitioning attributes, leaf-mildew and internal discoloration, that were assigned to the root node of the rule-tree. Different combinations of their values split the fifteen classes into five logically disjoint subsets. Four of these subsets correspond to single diseases: powdery-mildew, downy-mildew, charcoal-rot and brown-stem-rot, and the fifth one corresponds eleven diseases. For each of the eleven diseases, the program learned a ruleset distinguishing it from the other ones. As one can see in Figure 7, the rule-tree representation (on the left-hand side of the figure) appears simpler and easier to understand than the equivalent flat ruleset representation (on the right-hand side). Because AQ was applied to a smaller number of classes, the overall learning time of the rule-tree was shorter than the learning time of the flat ruleset.

5 AN EXAMPLE OF APPLICATION TO VOLCANOLOGY

This application was developed in collaboration with the Smithsonian Institution in Washington, D.C. Given a database with records of volcanic eruptions over the past 10,000 years provided by the Institution, the AQ21 learning system sought patterns in those eruptions. The data consisted of approximately 20,000 records characterizing individual cases of eruptions. Each case was described by 78 attributes of different types: binary, discrete, continuous, and structured (the domains of the structured attributes are hierarchies). A selection of these attributes used in this study is shown in Table 1.

Table 1: A selection attributes used in the Volcano Database.

Name	**Type**	**Description**
Subregion	Structured	Part of the world in which the volcano is located
Latx, Longx	Continuous	Latitude and longitude of the volcano
Upper	Discrete	Elevation of the peak (meters)
Upper1	Discrete	Height of the volcano (meters)
Type	Structured	Type of volcano
TC	Structured	Tectonic setting of the volcano
MapStatus2	Discrete	Indicates how long since last erupt (lower=more recent)
Year, Stop_year	Discrete	Years of eruption start and end, respectively
Radial_fissure	Binary	Whether there was a radial fissure eruption
Regional_fissure	Binary	Whether there was a regional fissure eruption
Island_forming	Binary	Whether the eruption resulted in the creation of an island
Subglacial	Binary	Whether there was a subglacial eruption
Crater_lake_erupt	Binary	Whether there was a crater lake eruption
Explosive	Binary	Whether the eruption was explosive in nature
Pyroclastic	Binary	Whether the eruption included pyroclastic materials
Lava_lake	Binary	Whether a lava lake was formed
Damage	Binary	Whether there was damage to human structures
Lahars	Structured	Whether lahars were formed
Tsunami	Binary	Whether the eruption resulted in a tsunami
Evacuation	Binary	Whether there were evacuations

NOTE: Attributes above the dashed line describe the volcano, and those below it describe individual eruptions.

In the experiment described here, the goal was to determine rules for differentiating eruptions in which fatalities were known to have occurred from eruptions without fatalities. The training set consisted of 50% of the cases of both types of eruptions randomly selected from the database; the remaining cases constituted the testing set. Figure 8 presents examples of rules learned for the two classes of eruptions.

[Fatalities = present]
  <= [Radial_fissure=present: 72,773] &
  [Tsunami=present: 61,29] &
  [Latx<=33.99: 343,5782] &
  [Stop_year<=1889: 148,934]: p=13, n=0; QUALITY=0.7

[Fatalities = absent]
  <= [Pyroclastic=absent: 8244, 221]: p=8244,n=221, QUALITY =0.4

Figure 8: Examples of discovered rules in the volcano database.

The first rule says that fatalities occurred in thirteen cases of eruptions with a radial fissure in which a tsunami occurred, and that the eruptions occurred south of 34 degrees north latitude, and ended prior to 1889. There were no exceptions to this rule. The second rule is a strong pattern that states that in 8244 out of 8472 documented cases, a non-pyroclastic eruption resulted in no fatalities. There were 221 exceptions to this pattern.

In all experiments, predictive accuracy of the discovered rules was greater than 90% on the testing dataset. One surprising result was that pattern sets in which inconsistency was permitted had a comparable predictive accuracy on the testing set as the complete and consistent rulesets, even though they were much simpler (involved far fewer rules and conditions). The generated rulesets were understandable and easy to interpret by the collaborating scientists from the Smithsonian Institution. This feature made it possible for them to adjust the rules that contained spurious conditions, and to improve the rules to reflect their expert knowledge (Kaufman and Michalski, 2006).

6 AN EXAMPLE OF APPLICATION TO DEMOGRAPHICS

These experiments sought to discover unknown patterns or anomalies in a dataset describing 190 countries in the CIA's World Factbook. Attributes describing countries included population growth rate, birth rate, death rate, net migration rate, fertility rate, infant mortality rate, literacy, life expectancy, and predominant religion.

This experiment involved conducting a *grand tour*, in which each attribute in the dataset is sequentially treated as an output (dependent) attribute, while the remaining ones are treated as input (independent) attributes. One example of a rule learned in the grand tour is a rule characterizing 25 of the 55 countries with low (<1%) population growth:

[PopGrRate < 1%]
<= [BirthRate = 10..20 or 50..60: 46, 20] &
  [FertRate = 1..2 or >7: 32, 17] &
  [Religion is Protestant or Catholic or Orthodox or Shinto: 38, 32] &
  [NetMigRate < +10: 54, 123]: p=25, n=0

This rule exposed an interesting anomaly. In its first condition, there is a range of birth rates from 50 to 60, which is rather high for cases with low population growth. Looking at the 25 countries that satisfied this rule, 24 had birth rates less than or equal to 20. Only one, Malawi, had a birth rate above 50. Investigating Malawi against the rest of the countries quickly revealed the reason for this surprising finding: the country had an outward net migration rate dwarfing those of all other countries in the world.

7 AN EXAMPLE OF APPLICATION TO INTRUSION DETECTION

The goal of these experiments was to discover patterns that characterize legitimate activities of computer users in order to detect computer intrusion or misuse. Such a misuse may be indicated when a purported user strongly violates the patterns characterizing his or her computer use. For this task, we developed a new methodology, called LUS (Learning User Signatures), that employs natural induction to derive models of users' behavior from the temporal datastreams characterizing their interaction with computers (Michalski et al. 2005; 2006).

The LUS methodology has several embodiments, depending on the type of user model is to be learned. Among the models we have explored are Multistate Templates, Prediction-based, Rule-Bayesian, and Activity-based models. Here we present briefly results using the multistate template model, which generates flexible templates that can concisely describe a large number of different user activities.

The learning of user models is a multi-step process that consists of extracting events from the system's process table, determining the most relevant attributes and the most relevant events in the training target dataset for each user, and applying a learning method, or a combination of learning methods under appropriate parameter settings to this dataset.

The methodology strives to develop user models that can be modified manually by human experts, and are able to reliably detect illegitimate user behavior from user data streams that are as short as possible. LUS strives to emulate several important aspects of human learning and recognition processes:

- Idiosyncracy: It searches for patterns that are most characteristic of a given user, so that recognition is possible from short episodes that contain such features.
- Satisfiability: If, at some point the observed behavior strongly matches one user model, and only weakly matches other models, the observation of the users' data stream stops, and a decision is reported.
- Understandability: It strives for creating user models that are easy to interpret and understand by humans.
- Incrementability: User models can be updated incrementally over time, without re-learning them from scratch.

The raw data stream (from the NJIT archive) comprised three sets of data consisting of records extracted from process tables. Each set contained records of 1282 sessions from 26 users. From the available data, we selected the 10 users that had the highest number of recorded sessions, and then extracted the first 10 sessions of each of these users for training, and the following 5 for testing.

The basic construct for learning multistate templates is the *n* x *k-gram*, or *multigram*, which is a generalization of the well-known concept of an *n*-gram. A multigram is an *n* x *k* matrix, consisting of *k* attributes whose values are recorded at *n* consecutive time instances. A multigram is a useful device for characterizing processes whose states at different time instances need to be described not by one, but by several attributes.

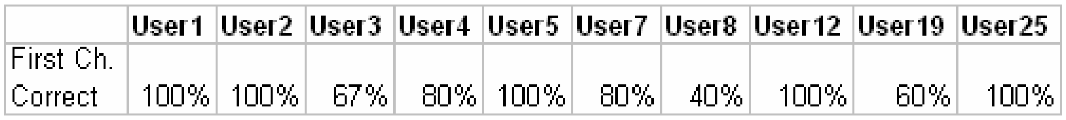
Figure 9 shows a multistate template learned from multigrams, with *n* set to 4. 4000 training multigrams represented User4's activities, and 25143 represented those of other users. The first condition in the template says that the hour in the third position of the multigram (the second most recent, since the last position is the most recent) must be between 11 and 14, i.e., that part of the activity took place between the 11 AM and 2 PM hours. That condition alone was satisfied by 3393 User4 multigrams and by 9171 of all other users. The second condition sets values on the process name at all four time positions of the multigram, and other conditions are interpreted similarly. It says, for example, that at the first time instance of the multigram the process name was netscape, outlook or winword. In all, this template was satisfied by 2419 multigrams of User4's activity, and none of other users.

```
 [User = user4]
<= [hour= << 11..14 : 3393,9171 (3) >> ] &
   [process_name = < netscape,outlook,winword : 3904,18376;
                     csrss,netscape,outlook,winword : 3909,18413;
                     csrss,netscape,outlook,winword : 3909,18397;
                     csrss,netscape,outlook,winword : 3909,18379 > ] &
   [event_status = < c,o : 3997,22090; c,o : 3997,22113; c,o : 3997,22123; * > ] &
   [proc_cpu_time_in_win_lf = < 0.3466..4.049 : 3611,12784; *; *; lt_3.916 : 3994,20119 > ] &
   [win_time_elapsed_lf = << gt_3.337 : 3251,13713 (1) >> ] &
   [delta_time_new_window = << lt_1800 : 3985,21445 (1) >> ] &
   [delta_time_new_window_lf = <<lt_7.748 : 3987,21518 (4) >> ] &
   [new_win_time_elapsed = <<300..18000 : 3954,16719 (4)>>] &
   [prot_words_chars = << lt_20 : 3980,17938 (1) >> ] &
   [proc_count_in_win_lf = << gt_4.063 : 3060,7992 (1) >>] &
   [win_opened_lf = < <1.498..2.636 : 3600,13531 (4) >> ]
       p = 2419, n = 0, P = 4000, N = 25173
```

Figure 9: A multistate template characterizing User4's activity.

During the course of this research, we tried many experiments with many different sets of parameters. In general, we used AQ21 to learn rules from which templates were formed, and then the EPIC program to test and classify. EPIC is an episode classifier; rather than classify individual events, it looks at the episode (a temporal sequence, typically a user session) as a whole, and sees which model best matches the entire episode.

Figure 10 shows the predictive accuracy of user models when they were matched against the testing data streams of the users (by "First Choice Correct" is meant the percentage of cases in which the degree of match between the testing data stream and the correct user model was the highest among all models).



	User1	User2	User3	User4	User5	User7	User8	User12	User19	User25
First Ch. Correct	100%	100%	67%	80%	100%	80%	40%	100%	60%	100%

Figure 10: First choice predictive accuracy of the user models.

Overall, 75% of the test episodes were correctly classified as EPIC's first choice. Whenever that was a reasonable similarity between training and testing data, the test episode was correctly classified. Of course, if no such similarity exists, no classifier will do well. Figure 11 presents a graphical representation how five of User25's testing episodes performed against the ten user models. The target model (User25) appears in black on the bar graphs. As one can see, the best match always indicates the correct user.

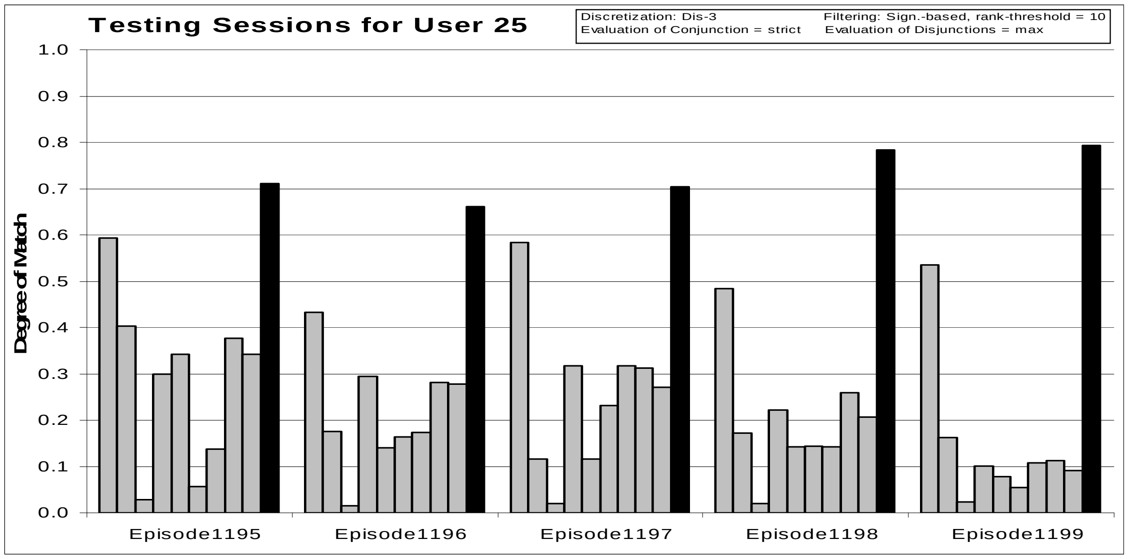


Figure 11: Classification of User25 test episodes.

Although these experiments have explored only a very small subspace of possible experiments on learning and testing of the developed user modeling methods, they show that the presented method can lead to an effective system for user modeling and intrusion detection under the following conditions:

- Sufficient training data for each user is available
- The target training data set for each user is appropriately determined
- The user's future behavior is "reasonably" similar to that recorded in the training data stream.

Details on this research are described in (Michalski et al. 2005).

8 AN EXAMPLE OF APPLICATION TO COMPLEX FUNCTION OPTIMIZATION

This section describes how natural induction was applied to guide evolutionary optimization of complex functions. The idea of guiding evolutionary optimization by machine learning has been embodied in Learnable Evolution Model (LEM), invented by Michalski; U.S. patent No 6,518988. The introductory paper on LEM (Michalski, 2000) is downloadable from *www.mli.gmu.edu*.

The novel idea in LEM is to apply machine learning at each stage of evolution to create hypotheses indicating subspaces of the search space that most likely contain the desirable solution. Instead of relying on blind operators of random mutations and/or recombinations, LEM applies hypothesis formation and instantiation operators, and thus constitutes a form of non-Darwinian evolutionary computation.

The newest implementation of LEM, called LEM3, employs AQ21 natural induction program for learning hypotheses, and was successfully applied to optimization of functions with the number of arguments (variables) ranging between 2 and 1000. In all our experiments on function optimization, LEM3 outperformed (sometimes by an order of magnitude or more) *every* tested method of conventional evolutionary computation in terms of the *evolution length*, measured by the number fitness function evaluations needed to achieve the same result (Wojtusiak and Michalski, 2006).

A particularly significant result of these experiments was that the LEM3's advantage over compared other evolutionary computation methods in terms of evolution length grew with the number of variables. This implies that LEM3 may be particularly advantageous for optimizing very complex functions.

A typical example of LEM3 performance is shown below, in which LEM3 was compared to EA, a standard evolutionary computation method, on the problem of optimizing the Rastrigin function of 500 variables. The Rastrigin function is defined by the equation:

$$f(x_1,\dots, x_n) = 10 * n + \sum_{i=1}^{n} (x_i^2 - 10 * \cos(2 * \pi * x_i))$$

The two dimensional case of the Rastrigin function (n=2) is illustrated on Figure 12.

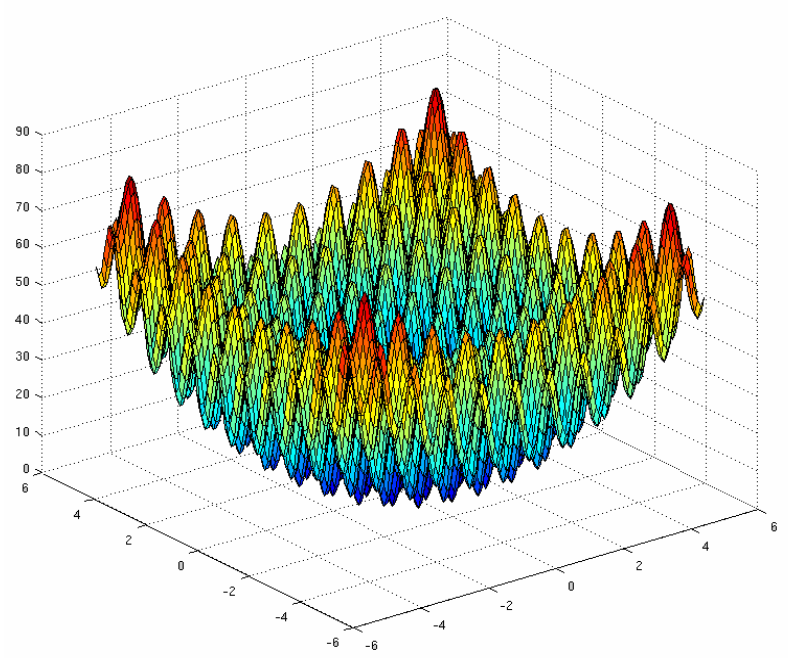


Figure 12: The Rastrigin function of 2 variables (from Wojtusiak and Michalski, 2005).

Figure 13 shows the performance of EA and LEM3. As one can see, results from EA and LEM3 were converging very fast toward the optimum (the value “0”) at the very beginning of the evolutionary computation. After this early phase, however, EA started to converge very slowly, while LEM3 continued to converge relatively fast. LEM3 reached the δ=0.1-close solution[3] after 5252 fitness function evaluations while EA reached the same solution after 128,184 evaluations (an average of 10 runs). The evolutionary speedup of LEM3 over EA for δ=0.1 was thus about 24.

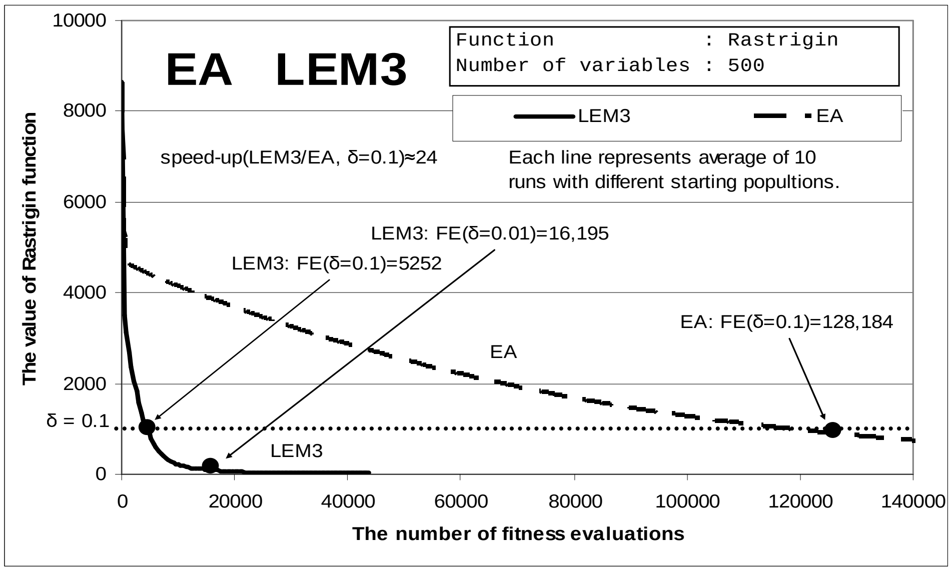


Figure 13: The LEM3 and EA evolutionary computation in minimizing the Rastrigin function of 500 variables (from Wojtusiak and Michalski, 2005).

For a detailed description of these experiments, see (Wojtusiak and Michalski, 2005; 2006).

9 AN EXAMPLE OF APPLICATION TO COMPLEX SYSTEM OPTIMIZATION

This example concerns the application of learnable evolution to the optimization of designs of heat exchangers used in refrigerators and air conditioners. A heat exchanger is a complex arrangement of tubes carrying a refrigerant (Figure 14). The optimization task is to configure the connection of tubes in a way that maximizes capacity of the heat exchanger for the technical parameters of the exchanger, such as number and configuration of tubes, and for the given environmental conditions.

This is a very complex, but practically very important optimization problem. Because of the ubiquity of heat exchangers, even a small improvement in their capacity may lead to huge economic and environmental benefits. For this optimization task, we developed a task-specific system, ISHED, that conducts optimization through evolutionary computation performed according to Learnable Evolution Model (LEM; Michalski, 2000).

[3] A δ-close solution is a distance between the solution and the optimal solution normalized by the value of the fitness function for the best individual in the initial population.

As mentioned earlier, LEM is a form of evolutionary computation in which the process of creating new individuals (here, designs) is guided by hypotheses created by a learning system. These hypotheses indicate the types of changes in the designs that will likely improve them.

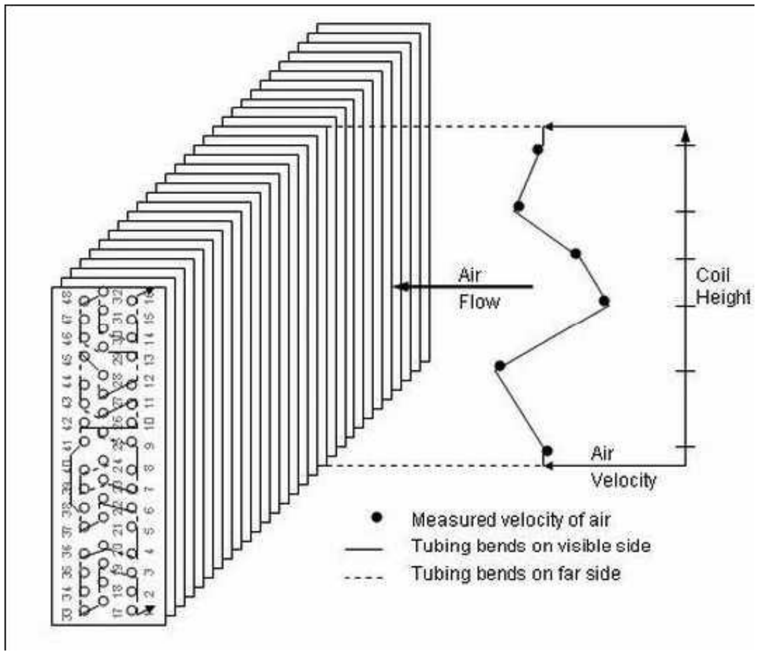

Air
Flow
Coil
Height
Air
Velocity
● Measured velocity of air
—— Tubing bends on visible side
---- Tubing bends on far side

Figure 14: An example of a heat exchanger.

This work has been conducted in collaboration with the National Institute of Standards and Technology (NIST). In experiments conducted by NIST, the ISHED system has generated designs that matched or outperformed the best human designs (Domanski et al., 2004). NIST is now trying to popularize ISHED among commercial companies.

10 AN EXAMPLE OF APPLICATION TO MANUFACTURING

The goal of these experiments was to assist a manufacturer in designing gearboxes meeting customer specifications. The customers provide specifications of the gearbox they need, which may include the size, mount, motor, flange, variable speed drive, gear ratio, and model number. Given such a specification, the manufacturer must either look up the components of that gearbox or, if no gearbox with those specifications has previously been developed, determine the components needed to build the requested gearbox (e.g., which shaft, which flange, which lubricant). This research developed an inductively learned rule base that provides assistance in this regard. It was conducted in collaboration with Lenze GmbH & Co KG, Germany.

Figure 15 shows examples of rules for selecting the Schnekenwelle (worm shaft) learned by AQ learning system (AQ19). For instance, the first rule says to use part number 00650943 for the worm shaft when the requested gear ratio is 5 and the model number is 104, 105, 113 or 145. Other rules are interpreted similarly.

[Schneckenwelle = 00650943]	<=	[Ratio = 5]	&	[Model=104,105,113,145]
00650944]	<=	[Ratio = 7]	&	[Model=104,105,113,145]
00650945]	<=	[Ratio = 10]	&	[Model=104,105,113,145]
00650946]	<=	[Ratio = 13]	&	[Model=104,105,113,145]
00650947]	<=	[Ratio = 15]	&	[Model=104,105,113,145]
00650948]	<=	[Ratio = 20]	&	[Model=104,105,113,145]
00650949]	<=	[Ratio = 26]	&	[Model=104,105,113,145]
00652155]	<=	[Ratio = 5]	&	[motor size = 80]
00652156]	<=	[Ratio = 7]	&	[motor size = 80]
00652157]	<=	[Ratio = 10]	&	[motor size = 80]
00652158]	<=	[Ratio = 13]	&	[motor size = 80]
00652159]	<=	[Ratio = 15]	&	[motor size = 80]
00652160]	<=	[Ratio = 20]	&	[motor size = 80]
00652161]	<=	[Ratio = 26]	&	[motor size = 80]
00652143]	<=	[Ratio = 5]	&	[motor size = 90]
00652144]	<=	[Ratio = 7]	&	[motor size = 90]
00652145]	<=	[Ratio = 10]	&	[motor size = 90]
00652146]	<=	[Ratio = 13]	&	[motor size = 90]
00652147]	<=	[Ratio = 15]	&	[motor size = 90]
00652148]	<=	[Ratio = 20]	&	[motor size = 90]
00652149]	<=	[Ratio = 26]	&	[motor size = 90]

Figure 15: Examples of discovered rules for assigning the correct *Schnekenwelle* (worm shaft) when given the gear ratio and model number or motor size.

Some of the discovered rules, such as those in Figure 15, were straightforward. For the requested size or gear ratio, certain parts were dictated. Other rules were more complex.

The rules learned by the program provided insights into the relationships and constraints of the gearbox manufacturing domain, and even exposed some errors in the data that had been provided. The learned classifier (a family of rulesets for different components) was able to select components with guaranteed 100% accuracy. Such high accuracy was possible because the data from which the classifier learned the rules included all known cases, and the natural induction program was run in the *theory formation mode* that generates descriptions that are fully complete and consistent with all the training data. This result was highly satisfactory for the Lenze company, not only because of the 100% accuracy of the rules, but also because they were so easy to interpret.

11 AN EXAMPLE OF APPLICATION TO CIVIL ENGINEERING

The research described in this section was conducted by Professor Kasperkiewicz and his team at the Institute of Fundamental Technological Research, Polish Academy of Sciences, in collaboration with the Machine Learning and Inference Laboratory.

One of the problems to which they applied our natural induction technology was to discover rules for distinguishing between different types of concrete, which are determined by the additives used in it. In the experiments described here, the goal was to identify concrete with silica fume as an additive. Results of the analysis were used to ascertain that silica fume was used as an additive as claimed.

The data was collected using acoustic emission sensors, and preprocessed using a wavelet transformation to obtain 10 input attributes: the nominal attribute “Class” with the domain {Cement Paste, Interface region, Aggregate, Void}, and nine numeric attributes such as the number of cases measured (Lzd), average energies (Sen), and average amplitudes (Saz) in three energy bands (H, M, L), and other. A decision attribute, “Composition,” has the three-valued domain {silica, no additives, pfa}. The training dataset consisted of 500 examples, with 302 examples in which silica was present, and the rest in which silica was not present.

Here is one of the strong patterns discovered by the AQ learning program (version AQ19) in this dataset:

[Composition=silica] <= [IzdM=23..233.5] & [SazM<28]

stating that concrete with silica as an additive is indicated if the number of events medium level (LzdM) is between 23 and 233.5, and the average amplitude of medium level signals (SazM) is smaller than 28.

This result was determined by the expert to be “more suitable” for use than the one obtained by the well-known and widely used commercial learning program See5 (Kasperkiewicz, 2005). Professor Kasperkiewicz and his research team have also applied the AQ natural induction method to several other civil engineering problems (e.g., Kasperkiewicz, 2003).

12 AN EXAMPLE OF APPLICATION OF CONCEPTUAL CLUSTERING TO AGRICULTURE

An important question in the research on unsupervised learning is how well classifications discovered by a learning program correspond to categorizations developed by experts. This problem was investigated by applying conceptual clustering program CLUSTER/2 developed in our laboratory and 18 other numerical taxonomy techniques (implemented in the NUMAX system) to an unclassified version of the soybean disease data that was the basis for the experiments presented in Section 4. The goal of the experiments was to see if the clustering program would re-create a classification of four selected diseases, Diaporthe stem canker (denoted D1), Charcoal rot (D2), Rhizoctonia root rot (D3), or Phytophtora rot (D4) on the basis on examples of these diseases without telling the program which disease these examples represent. The experiment used 47 cases of the above diseases, described by 35 mutli-valued attributes.

Only 4 of the 18 taxonomies created by the NUMAX program matched exactly the correct classification. None of the techniques used by the program provided any description of the created classes.

CLUSTER/2 reconstructed the classification of the diseases without a single error, and provided descriptions of individual classes. For example, Figure 16(a), shows the description of the cluster generated by CLUSTER/2 that corresponds to D1. Figure 16(b) shows a plant pathologist’s description of the symptoms of D1, which is called by experts Diaporthe stem canker.

As one can see from Figures 16 (a) and (b), the program-generated description for cluster D1 corresponds well to the expert description. The program’s description contains all the symptoms specified by the expert, plus additional conditions that were added as they were observed in the input data.

[Precipitation = Above normal] &	*[Precipitation = Normal or Above] &*
[Temperature = Normal] &	*[Temperature = Normal or Above] &*
[Stem Cankers = Above 2nd node] &	*[Stem Cankers = Above 2nd node] &*
[Canker Lesion Color = Brown or N/A] &	*[Canker Lesion Color = Brown] &*
[Fruiting Bodies = Present] &	*[Fruiting Bodies = Present] &*
[Condition of Fruit Pods = Normal] &	*[Condition of Fruit Pods = Normal] &*
[Time of Occurrence = Jul-Oct] &	*[Time of occurrence = Aug-Sep]*
[Damaged Area = Scattered or Low] &	
[Severity = Potential or Severe] &	
[Seed Treatment = None or Fungicide] &	
[Plant Height = Abnormal] &	
[Condition of Leaves = Abnormal] &	
[Leaf Spots = Absent] &	
[Shotholing/Shreading = Absent] &	
[Leaf Malformation = Absent] &	
[Leaf Mildew Growth = Absent] &	
[Condition of Seed = Normal] &	
[Condition of Stem = Abnormal] &	
[Extern. Stem Decay = Firm and Dry] &	
[Mycelium on Stem = Absent] &	
[Int. Discolor of Stem = None] &	
[Sclerotia int. or ext. = Absent] &	
[Condition of Roots = Normal] &	
[Plant stand = Normal]	
(a) A CLUSTER/2-generated description	(b) An expert-provided description

Figure 16: Two descriptions of Diaporthe stem canker in soybeans.

This is a very satisfactory result, because it shows that the conceptual clustering program not only reconstructed correctly experts' classification of the diseases on the basis of a limited sample of data, but also created descriptions of diseases that match well their expert-provided characterizations. For more details on this research and on conceptual clustering, consult (Michalski and Stepp, 1983a,b; Seeman and Michalski, 2006).

13 AN EXAMPLE OF APPLICATION TO MUSICOLOGY

This section describes an application of conceptual clustering to determining a classification of Spanish songs. The dataset, provided by musicologist Pablo Poveda (Poveda, 1980), consists of descriptions of 100 Spanish songs in terms of 22 attributes, some are binary, some multi-valued, and some continuous. The clustering evaluation criterion was to seek clusters whose description had minimum *sparseness*, (defined as the ratio of the number of examples covered the description and the number of all possible examples that could satisify this description). The taxonomy automatically generated by the conceptual clustering program is shown in Figure 17.

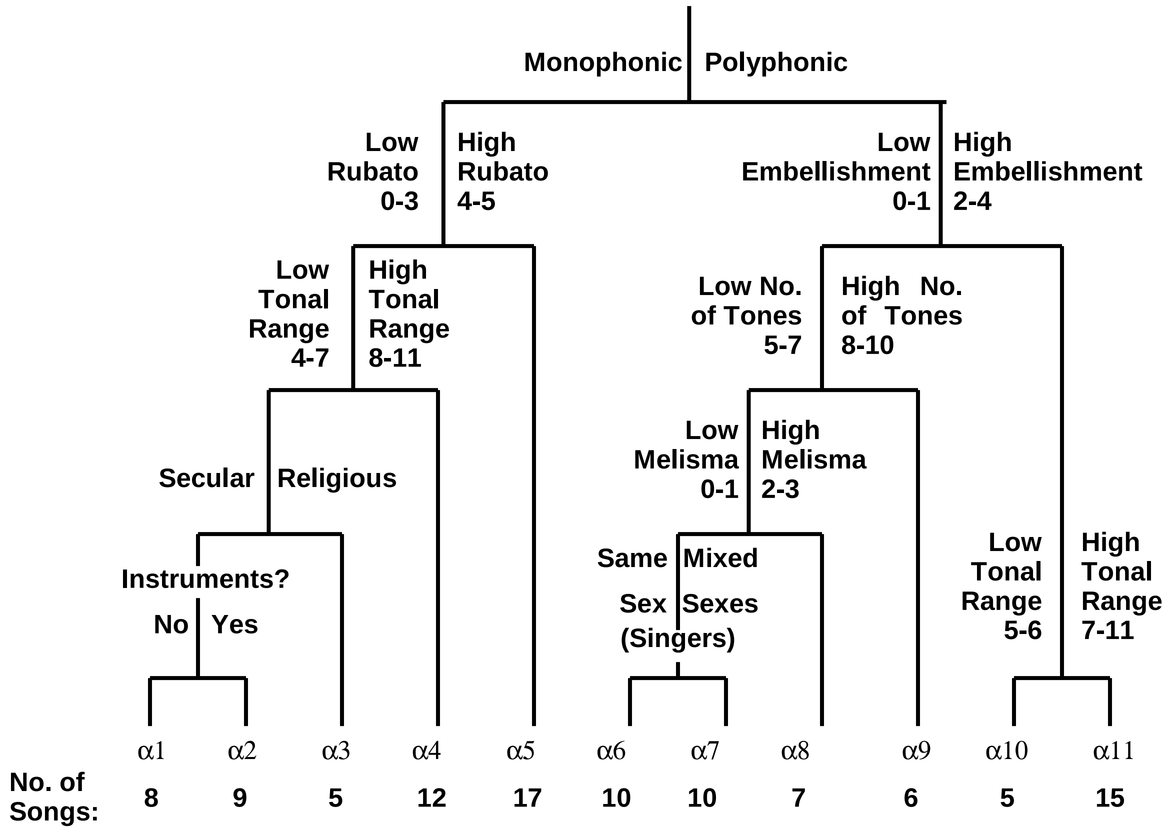


Figure 17: A classification hierarchy of Spanish folk songs produced by conceptual clustering (figure reproduced from Michalski and Stepp, 1983a).

At the top level, the songs are divided into monophonic and polyphonic harmonic structures. The monophonic songs are then divided into those with low or high degrees of rubato, while the polyphonic ones are subdivided according to their degrees of embellishment. The full taxonomy divides the songs into a total of 11 classes, each consisting of between 5 and 17 songs, as indicated by numbers associated with the leaves of the hierarchy. By tracing branches from the root to the leaves, one can create a description of classes of songs associated with different leaves.

This result was highly evaluated by the musicologist, because it corresponds well to his own sense as to how best to classify the songs. He was particularly pleased by the fact that he could see an understandable description of each class of songs generated by the program, in contrast to the taxonomy he had obtained using a conventional, similarity-based clustering program that did not provide such descriptions.

14 AN EXAMPLE OF APPLICATION TO TAX FRAUD DETECTION

In these experiments, done by our student Scott Fischthal at Lockheed Martin Corporation, conceptual clustering and natural induction were combined to develop rules for detecting tax fraud. In this approach, data (tax returns) were grouped by a conceptual clustering program, and then supervised learning was applied to examples of known fraud in .each group, in order to generate simple rules distinguishing regular and fraudulent tax forms within each group. These rules were finally applied to new tax returns. Figure 18 depicts this methodology.

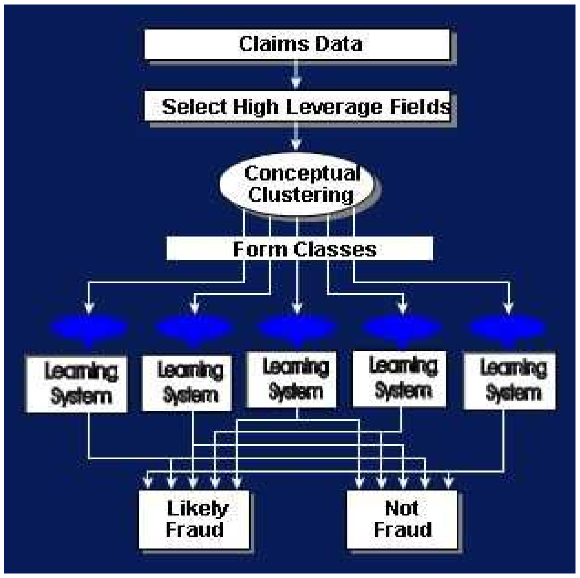


Figure 18: Clustering and rule learning for tax fraud detection (the diagram made by Scott Fischthal).

These experiments created different groups of taxpayers, and among them found a cluster with a much higher percentage of tax violators than in other clusters (Figure 19).

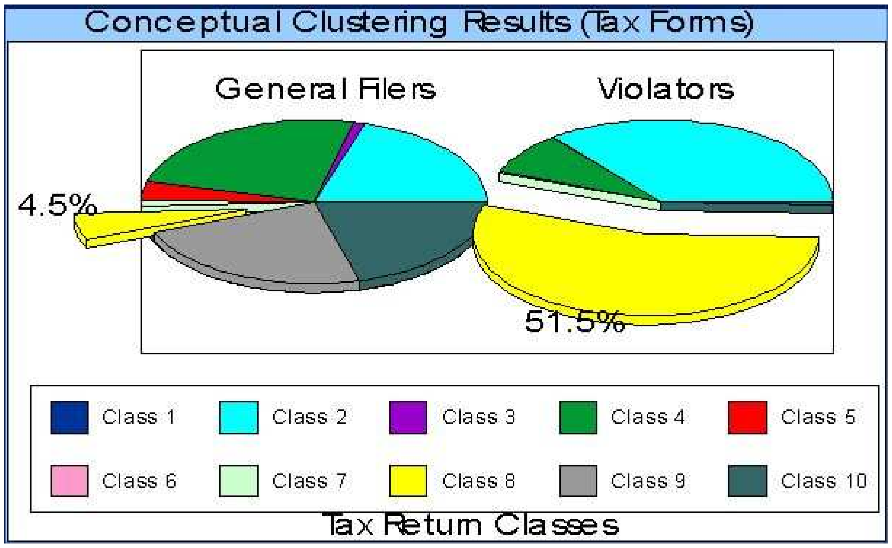


Figure 19: Distribution of filers and cases of fraud among the discovered groups (result by Scott Fischthal).

The above figure shows that taxpayers with the profile satisfying the description of the discovered cluster on the right of the figure are much more likely to submit a fraudulent tax form return than those that do not satisfy that description.

15 COMPARING NATURAL INDUCTION WITH OTHER METHODS ON A SIMPLE DESIGN PROBLEM

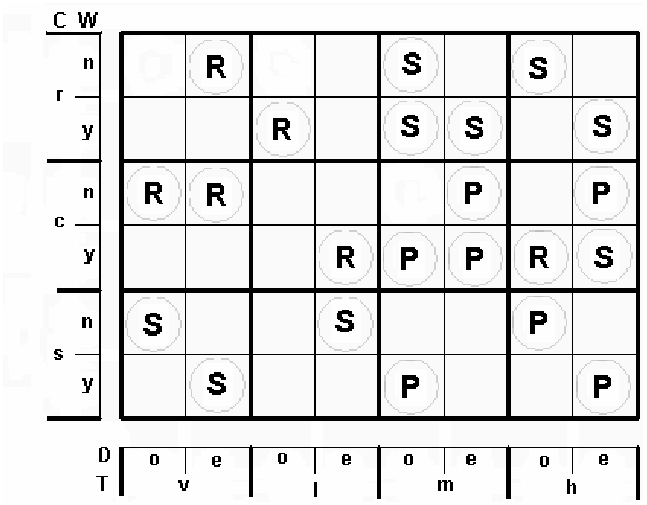
15.1 Problem Definition

In order to help the Reader get a quick insight into the natural induction capabilities of the AQ21 program and those of some other well-known learning programs, we designed a simple, easy to understand problem. In this problem, given entities are described in terms of the attributes presented in Figure 20.

Attribute	Type	Domain
Condition	**discrete**	{rain, cloudy, sunny}
Wind	**nominal**	{no, yes}
Temperature	**discrete**	{very_low, low, medium, high}
Daytype	**nominal**	{workday, weekend}
Activity	**nominal**	{Play, Shop, Read}

Figure 20: Attributes, their types, and their domains.

"Activity" is an output attribute that characterizes the activity of a person during a given time interval, and is assumed to be a function of the remaining (input) attributes. Suppose that our task is to discover a general rule characterizing the dependence of the activity "Play" on the input attributes, on the basis of examples that link different activities to the sets of input attribute values (training examples). Figure 21 presents a General Logic Diagram spanned over the input attributes and the examples of three classes P (Play), S (Shop), and R (Read).



C – Condition, W – Wind, T – Temperature, D – Daytype, r – rain, c – cloudy, s – sunny, n – no, y – yes, v – very low, l – low m – medium, h – high, o – workday, e – weekend, P – play, R – read, S - shop

Figure 21: A GLD with 22 examples of different values of the Activity attribute.

Each cell of the diagram corresponds to one combination of input attribute values (an *event*). Training examples are represented by placing in the cells the first letter of the activity associated with the events the cells represent. Empty cells indicate events for which activity is unknown and will be hypothesized through learning.

15.2 Solutions to the Problem by Different Learning Programs

To the problem described above, we applied the AQ21 program and some well-known programs, specifically, C4.5 (Quinlan, 1993), RIPPER (Cohen, 1995), and CN2 (Clark and Niblett, 1989). The results are as follows.

Figure 22 presents a decision tree and Figure 23 a set of rules determined by C4.5. Both the decision tree and the rules were partially inconsistent with the training data. In the case of decision rules, the activity "Play" is defined partially explicitly and partially implicitly. The implicit value "Play" is a default decision that is assigned when the event to be classified does not satisfy the conditions stated explicitly.

```
Condition = rain: shop (7.0/3.4)
Condition = cloudy:
|   Temperature = very_low: read (2.0/1.0)
|   Temperature = low: read (1.0/0.8)
|   Temperature = medium: play (3.0/1.1)
|   Temperature = high: play (3.0/2.8)
Condition = sunny:
|   Temperature = very_low: shop (2.0/1.0)
|   Temperature = low: shop (1.0/0.8)
|   Temperature = medium: play (1.0/0.8)
|   Temperature = high: play (2.0/1.0)
```

Figure 22: A decision tree learned by C4.5.

```
Condition = cloudy & temperature = medium
->  class play  [63.0%]
Condition = sunny & temperature = high
->  class play  [50.0%]
Condition = rain
->  class shop  [51.2%]
Condition = sunny & temperature = very_low
->  class shop  [50.0%]
Temperature = very_low
->  class read  [35.2%]
Temperature = low
->  class read  [31.4%]
Default class: play.
```

Figure 23: Decision rules derived by C4.5 from the C4.5 decision tree.

The RIPPER program applied to the same dataset determined the rules presented in Figure 24. These rules need to be evaluated sequentially. For example, to determine the activity "Play," it is first necessary to evaluate two rules for activity "Read", and if they are not satisfied, then "Play" is assigned.

```
read :- Temperature=very_low (3/2).
read :- Temperature=low (2/1).
play :- Condition=sunny (3/0).
play :- Condition=cloudy, Wind=no (2/0).
Play :- Condition=cloudy,Temperature=medium (2/0)
default shop (6/1).
```

Figure 24: Rules learned by RIPPER.

Decision rules determined by the CN2 program are presented in Figure *25*

```
IF    Condition = cloudy AND temperature = medium
THEN  Activity = play  [3 0 0]
IF    Condition = sunny AND temperature = high
THEN  Activity = play  [2 0 0]
IF    Condition = sunny AND temperature = medium
THEN  Activity = play  [1 0 0]
IF    Wind = no AND temperature = high AND
      Daytype =weekend THEN activity =play[1 0 0]
IF    Condition = rain AND temperature = medium
THEN  Activity = shop  [0 3 0]
IF    Condition = sunny AND temp = very_low
THEN  Activity = shop  [0 2 0]
IF    Condition = rain AND temperature = high
THEN  Activity = shop  [0 2 0]
IF    Wind = no AND temperature = low
THEN  Activity = shop  [0 1 0]
IF    Condition = cloudy AND wind = yes
  AND Temperature = high AND daytype = weekend
THEN  Activity = shop  [0 1 0]
IF    Condition = cloudy
  AND Temperature =very_low
THEN  Activity = read  [0 0 2]
IF    Wind = yes AND temperature = low
THEN  Activity = read  [0 0 2]
IF    Condition = rain AND temperature = very_low
THEN  Activity = read  [0 0 1]
IF    Wind = yes AND temperature = high
      AND Daytype = workday
THEN  Activity = read  [0 0 1]
(DEFAULT) Activity = shop  [7 9 6]
```

Figure 25: Rules determined by CN2.

The subsequent Figures 26, 27, 28, and 29 present different types of rules learned by AQ21. Section 14.4 summarizes the predictive accuracy of the descriptions generated by different programs for the training and testing data sets.

In these experiments, the testing set consisted of all events that are represented by empty cells in Figure 21 (events with unknown classification). The predictive accuracy on the testing set thus evaluates the quality of generalization performed by the different programs.

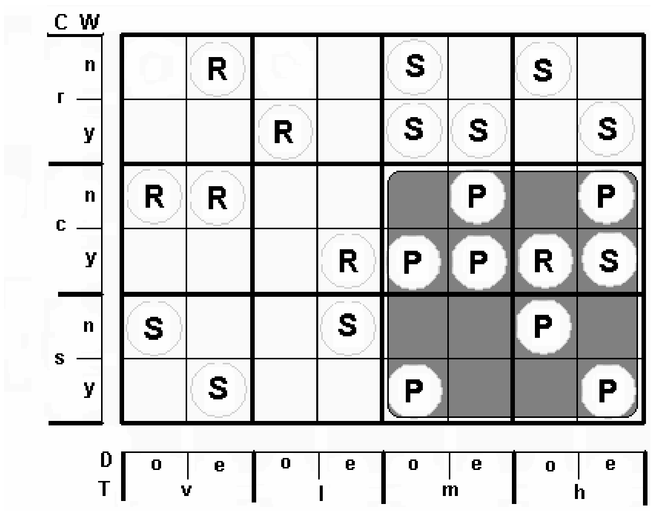
The presented results from AQ21 were obtained by requesting from it different types of descriptions. When instructed to determine strong patterns for the Activity "Play," AQ21 determined the pattern presented in Figure 26.

```
[Activity=play]
    <=  [Condition=cloudy v sunny: 7,8] &
        [Temperature=medium v high: 7,7]:
        p=7,n=2,QUALITY=0.67
```

Figure 26: A strong pattern for Activity "Play" determined by AQ21.

The pattern consists of a single attributional rule stating that the activity is "Play" if the weather is cloudy or sunny, and the temperature is medium or high. The rule covers 7 positive and 2 negative examples, and its quality, q(*w*), is 0.67, where *w* takes the default value 0.5 (which requests putting an equal emphasis on completeness and confidence criteria).

Numbers inside conditions represent positive and negative coverages of the conditions, considered individually. The pattern is graphically illustrated in Figure 27.



C – Condition, W – Wind, T – Temperature, D – Daytype, r – rain, c – cloudy, s – sunny, n – no, y – yes, v – very low, l – low, m – medium, h – high, o – workday, e – weekend, P – play, R – read, S - shop

Figure 27: GLD showing the strong pattern discovered by AQ21.

As one can see in Figure 21, two examples included in the pattern (one R and one S) do not represent "Play" activity. They are exceptions from this simple pattern.

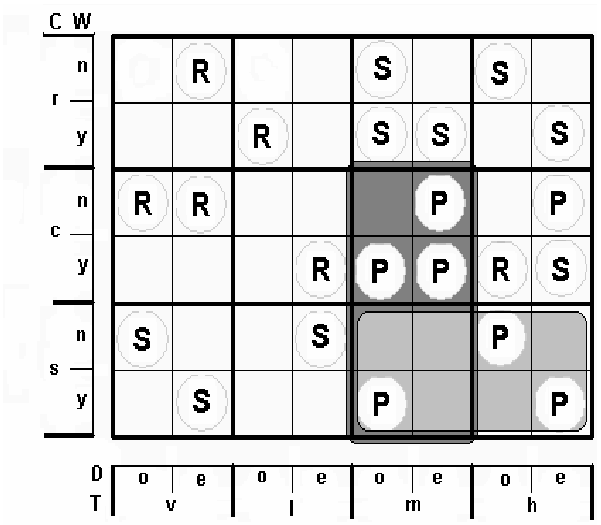
AQ21 allows the user to control the tradeoff between completeness and confidence of patterns by adjusting parameter *w* in the pattern quality measure q(*w*) (Michalski and Kaufman, 2001). When setting the *w* parameter to 0.15, AQ21 found two rules presented in Figure 28 and graphically in Figure 29. The two-rule pattern is consistent with data but incomplete, because one training example, the (c,h,n,e) example as seen in Figure 29, is not covered by the learned rules.

It may be illustrative to note that by reducing *w* to 0, a complete and consistent ruleset would be obtained. Such a ruleset could be directly obtained by executing AQ21 in Theory Formation mode (rather than Pattern Discovery mode, used here).

```
[Activity= Play]
    <=  [Condition=cloudy v sunny: 7,8] &
        [Temperature=medium: 4,3]:
        p=4,n=0,QUALITY=0.919

    <=  [Condition=sunny: 3,3] &
        [Temperature=medium v high:7,7]:
        p=3,n=0,QUALITY=0.881
```

Figure 28: Rules hypothesized by AQ21 for *w* = .15.



C – Condition, W – Wind, T – Temperature, D – Daytype, r – rain, c – cloudy, s – sunny, n – no, y – yes, v – very low, l – low, m – medium, h – high, o – workday, e – weekend, P – Play, R – Read, S - Shop

Figure 29: A visualization of AQ21 rules presented in Figure 28 using General Logic Diagram.

15.3 Learning Rules with Exceptions

The concept of an "exception" is commonly used by people when describing anomalies rarely occurring in comparison to other features in phenomena being described. It is not unusual that a simple theory may work well for most cases, but turning it into a fully consistent and complete theory would require making it significantly more complex. In such cases, it is desirable to learn rules with exceptions (e.g., Michalski, 2004; Yao et al, 2004). AQ2*1* can be set to learn rules with exceptions in the following form:

CONSEQUENT <= PREMISE |_ EXCEPTION

where EXCEPTION is either an attributional conjunctive description, or a list of examples constituting exceptions to the rule. Note that exceptions in such rules are always negative examples that haven been included in PREMISE.

The processes of learning rules with exceptions in Theory Formation and Pattern Discovery modes are different. In the latter mode, where inconsistency is allowed, the program learns standard patterns and generates exceptions representing covered negative examples, by finding a conjunctive description of the negative examples using AQ learning.

In Theory Formation mode, where consistency is guaranteed, the program adds negative examples to the list of exceptions if such examples are infrequent, but would introduce significant complexity to a description that does not cover them. If all of the exceptions can be characterized by one conjunctive description, such a description is used as the exception clause in the rule; otherwise, an explicit list of exceptions is outputted.

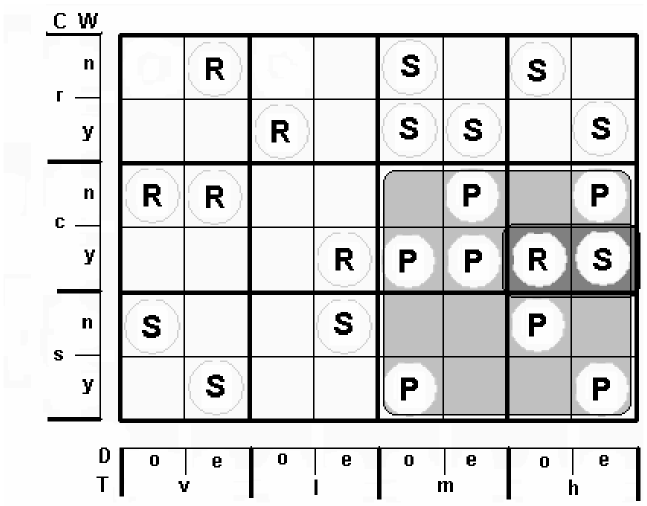
AQ21 applied to the same data as above produced the rule with EXCEPTION presented in Figure 30.

```
[Activity=Play]
     <=  [Condition=cloudy v sunny: 7,8] &
         [Temperature=medium v high: 7,7]
     |_  [Condition=cloudy] & [Wind=yes] & [Temperature=high]
         :p=7,n=0,QUALITY=1
```

Figure 30: Strong pattern with exception found by AQ21.

The rule states that activity is "Play" if weather is cloudy or sunny and temperature is medium or high, unless weather is cloudy, there is wind, and temperature is high.

Figure 31 shows a generalized logic diagram (GLD) representation of the rule presented in Figure 30. The two highlighted examples representing "Shop" (S) and "Read" (R) activities are treated as exceptions to the general pattern for the play activity.



C – Condition, W – Wind, T – Temperature, D – Datyime, r – rain, c – cloudy, s – sunny, n – no, y – yes, v – very low, l – low, m – medium, h – high, o – workday, e – weekend, P – Play, R – Read, S - Shop

Figure 31: GLD showing the strong pattern with exception found by AQ21.

AQ*21* generalized the two exceptions into one conjunctive description:

[Weather=cloudy] & [Wind=yes] & [Temperature=high].

In the case that the two examples could not be generalized into one conjunctive description, the program would have listed them explicitly:

```
cloudy, yes, high, yes,   shop
cloudy, yes, high, no,    home.
```

This simple example shows that the introduction of rules with exceptions may produce descriptions that are simpler, more accurate, and more comprehensible than descriptions without exceptions.

15.4 Discussion of Results

This discussion evaluates results obtained by different programs in terms of two criteria: the simplicity of the descriptions learned and the error rate on the training data (there was no testing data in this problem).

The C4.5 program generated a decision tree with 3 internal nodes and 11 branches from the given training data (22 examples). When applied to this data, it made 4 classification errors (18%). The C4.5 rules (7 rules), generated from the decision tree, made 5 classification errors (23%) on the training dataset.

RIPPER generated 6 rules, which made also 4 errors (18%) on the training data. CN2 generated 13 rules that were complete and consistent with the data (no errors on the training dataset), but constituted a much more complex description (13 rules).

AQ21, run in Pattern Discovery mode (using the default value of w=0.5), generated only one strong pattern that covered all positive examples (a complete description), but also covered 2 examples of other classes (9% error on the training set). When executed with w=0.15, AQ21 produced a consistent description and nearly-complete, as it missed only one example of activity "Play" (4.5% error). When run in Theory Formation mode, AQ21 generated three simple rules (as compared to 11 rules learned by CN2), which constituted a complete and consistent generalized description of the training data (0% errors).

When run in the mode generating censored rules (rules with an exception clause), AQ21 discovered one censored rule that was also complete and consistent with regard to the training data (0% errors). None of the compared programs, except AQ21, had the ability to determine patterns describing only one class, but always generate a description for all classes.

The accuracies of the descriptions obtained by different methods are summarized in Table 2. As one can see, in this simple example, AQ21 produced both more accurate and simpler concept descriptions or patterns than the other programs. These results can be attributed to the richer representation language that is used by this program. Using more expressive representation language makes possible for the program to generate simpler hypotheses and also in the forms easier to understand and interpret in natural language. AQ21 also allows the user to control the type of description to be learned, by choosing the mode of program operation and by appropriately setting the parameter w in the measure of description QUALITY. The latter determines the relative importance given to coverage and confidence of the rules to be learned.

Table 2: Comparison of the performance of different learning methods.

	Training		**Testing**	
	"Play"	All Classes	Play"	All Classes
C4.5	*100%*	*81.8%*	*100%*	*76.9%*
C4.5Rules	*100%*	*77.3%*	*100%*	*65.4%*
RIPPER	*100%*	*77.3%*	*100%*	*80.8%*
CN2	*100%*	*100%*	*85.8%*	*80.8%*
AQ21 PD Play	*100%*	*90.91%*	*100%*	*100%*
AQ21 Ex. Play	*100%*	*100%*	*100%*	*100%*
AQ21 Ex. All	*100%*	*90.91%**	*100%*	*100%*
AQ21 PD All	*100%*	*90.91%*	*100%*	*100%*
AQ21 TF	*100%*	*100%*	*100%***	*100% ****

* 2 events (exceptions) classified as "do not know" (which is treated by the current ATEST testing program as wrong classification)
** with Precision of 81.25%
*** with Precision of 89.29%
(Precision less than 100% indicates that another class was also indicated; Wojtusiak, 2004)

Although it was not mentioned in the paper before, AQ21 also allows the user to define a multi-criterion measure of the description optimality, which the user can assemble using predefined elementary criteria. This feature can be useful when different criteria of description optimality are most suitable for different problems. In the described experiments, we used a default setting of this measure.

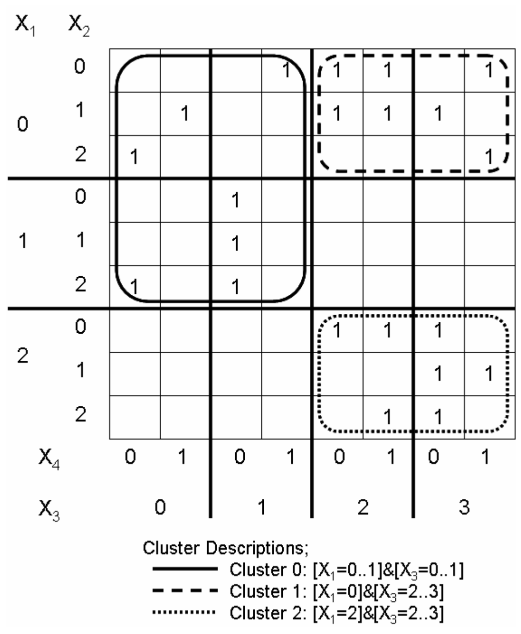
16 COMPARING CONCEPTUAL CLUSTERING WITH CONVENTIONAL CLUSTERING ON A SIMPLE DESIGN PROBLEM

To give the reader an insight into the differences between conceptual clustering and a conventional, similarity-based clustering, this section describes an application of both methods to a very simple designed problem (it is based on Seeman and Michalski, 2006).

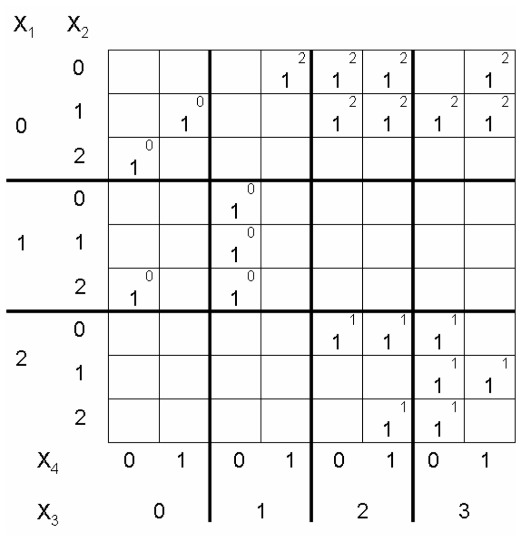
The objects in the dataset to be clustered are described by the following four multi-valued attributes, with domains denoted by {}: X1: {0,1,2}, X2: {0,1,2}, X3: {0,1,2,3}, and X4: {0,1}. The dataset consists of 21 tuples (vectors of attribute values) that are represented by a "1" in the diagram in Figure 32(a).

The dataset was clustered by CLUSTER3, our newest implementation of conceptual clustering, and by the KMlocal program that implements Lloyd's conventional clustering algorithm (Kanugo et al., 2002). The Lloyd's algorithm assigns observations to clusters using the minimum Euclidean distance between the observation and the cluster centroids. Both KMlocal and CLUSTER3 were run with default parameters. The number of clusters was set to 3 for both programs.

The three simple disjoint cluster descriptions produced by CLUSTER3 are listed, and represented visually by the GLD in Figure 32(a). A similar GLD in Figure 32(b) indicates the results of the KMlocal application. In Figure 32(a), ellipses indicate the conceptual descriptions of the clusters; no ellipses are shown in Figure 32(b), because KMlocal creates groups of datapoints, but does not describe them. In Figure 32(b), the number of the cluster to which events have been assigned is indicated in the upper right-hand corner of the cells representing events in the data.


X_1 X_2
X_4
X_3
Cluster Descriptions;
Cluster 0: [X_1=0..1]&[X_3=0..1]
Cluster 1: [X_1=0]&[X_3=2..3]
Cluster 2: [X_1=2]&[X_3=2..3]

(a) Clusters generated by CLUSTER3



(b) Clusters generated by KMlocal

Figure 32: A GLD representation of clusters of the designed dataset.

As seen in Figure 32, clusters produced by KMlocal and CLUSTER3 are similar, except for the event (0,0,1,1) which was assigned to cluster 0 by CLUSTER3, and to cluster 2 by KMlocal. The difference in the assignment was due to the value of X_2 for this event. CLUSTER3 included this event in cluster 0, because it can be described together with other events of cluster 0 by one simple description (X1 = 0 or 1, and X3 = 0 or 1). The simplicity of this description was considered more important than a small difference in the Euclidean distance.

This above illustrates two important differences between CLUSTER3 and KMlocal in that the former produces descriptions of clusters it generates, while the latter does not, and that properties of the description (e.g., its simplicity and the sparseness of events in it) is taken into consideration when creating clusters.

Another difference is that descriptions produced by CLUSTER3 are generalizations of events, in the sense that they not only cover the events in the dataset, but also cover unobserved events. This way, new events can be easily classified to an appropriate category simply by determining which description it matches. For example, the event (0,2,1,1), not present in the data, would be classified to Cluster 0 because it satisfies the description of that cluster. In contrast, in the case of KMlocal, to determine whether this event should be classified to Cluster 0 or to Cluster 2 would require a re-execution of the KMlocal upon the entire dataset.

17 CONCLUSION

This paper reviewed examples of application of natural induction and conceptual clustering to a wide range of real-world problems. The presented results show that in every application, natural induction was able to hypothesize general descriptions or patterns in data that had high predictive accuracy and were also simple and easy to interpret.

The differences between natural induction and several well-known methods were illuminated by comparing the method of natural induction implemented in the AQ21 program with several well-known methods on a simple designed problem. The final section also provided a very simple illustration of the differences between conceptual clustering and conventional similarity-based clustering.

For technical details on the methods of natural induction and conceptual clustering employed in the applications described here, visit http://www.mli.gmu.edu. This site has many papers on these methods and their various implementations. It also contains downloadable program AQ21. For the current version of program CLUSTER3, which is still under development, contact MLI system manager Janusz Wojtusiak (jwojt@mli.gmu.edu), or Ryszard Michalski, PI of the grants that supported this research (michalski@mli.gmu.edu).

By demonstrating the applicability of the described methods to diverse real-world problems, we hope to invoke Readers’ interest in applying these methods to problems in their own fields.

REFERENCES

Baim, P., “The PROMISE Method for Selecting Most Relevant Attributes For Inductive Learning Systems,” *Reports of the Intelligent Systems Group*, ISG 82-1, UIUCDCS-F-82-898, Department of Computer Science, University of Illinois, Urbana, September, 1982.

Clark, P. and Niblett, T., “The CN2 Induction Algorithm,” *Machine Learning* 3: pp. 261-289, 1989.

Cohen, W., “Fast Effective Rule Induction,” *Proceedings of the 12th International Conference on Machine Learning*, 1995.

Domanski, P.A., Yashar, D., Kaufman, K. and Michalski, R.S., “An Optimized Design of Finned-Tube Evaporators Using the Learnable Evolution Model,” *International Journal of Heating, Ventilating, Air-Conditioning and Refrigerating Research*, 10, pp. 201-211, April, 2004.

Kasperkiewicz J., **“**Artificial Intelligence Methods and Analysis of Structure in Evaluation of Hardened Concrete Quality,” *RILEM 2nd International Workshop on Life Prediction and Aging Management of Concrete Structure,* Paris, France, pp.185-194, May, 2003.

Kasperkiewicz J., “On a Possibility of Structure Identification by Microindentation and Acoustic Emission,” *Proceedings of the 2nd International Symposium on Nanotechnology in Construction*, Bilbao, Spain, November, 2005.

Kaufman, K.A. and Michalski, R.S., “Determining Patterns in the Database of Volcanic Eruptions Using Natural Induction,” 2006 (in preparation).

MacDonald, T.J., Brown, K., LaFleur, B., Paterson, K., Lawlor, C., Chen, Y., Packer, R., Cogen, P. and Stephan, D., “Expression Profiling of Medulloblastoma: PDGFRA and the RAS/MAPK Pathway as Therapeutic Targets for Metastatic Disease,” *Nature Genetics*. 29, pp. 143-152, October 2001.

Michalski, R.S., “A Variable-Valued Logic System as Applied to Picture Description and Recognition,” in F. Nake and A. Rosenfeld (eds.), *Graphic Languages,* North-Holland Publishing Co., 1972.

Michalski, R.S., “LEARNABLE EVOLUTION MODEL Evolutionary Processes Guided by Machine Learning,” *Machine Learning*, 38, pp 9-40, 2000.

Michalski, R.S., “Attributional Ruletrees: A New Representation for AQ Learning,” *Reports of the Machine Learning and Inference Laboratory*, MLI 02-1, George Mason University, Fairfax, VA, October, 2002.

Michalski, R.S., “ATTRIBUTIONAL CALCULUS: A Logic and Representation Language for Natural Induction,” *Reports of the Machine Learning and Inference Laboratory*, MLI 04-2, George Mason University, Fairfax, VA, April, 2004.

Michalski, R.S. and Kaufman, K., “Learning Patterns in Noisy Data: The AQ Approach,” in G. Paliouras, V. Karkaletsis and C. Spyropoulos (Eds.), *Machine Learning and its Applications,* Springer-Verlag, pp. 22-38, 2001.

Michalski, R.S., Kaufman, K., Pietrzykowski, J., Sniezynski, B. and Wojtusiak, J., “Learning User Models for Computer Intrusion Detection: Preliminary Results from Natural Induction Approach,” *Reports of the Machine Learning and Inference Laboratory*, MLI 05-3, George Mason University, Fairfax, VA, November, 2005.

Michalski, R.S., Kaufman, K., Pietrzykowski, J., Sniezynski, B. and Wojtusiak, J., “Learning Symbolic User Models for Intrusion Detection: A Method and Initial Results,” *Proceedings of the Intelligent Information Processing and Web Mining Conference, IIPWM 06*, Ustron, Poland, June 19-22, 2006.

Michalski, R.S. and Stepp, R.., “Learning from Observation: Conceptual Clustering,” in R.S. Michalski, J. Carbonell and T.J. Mitchell (eds.),.*Machine Learning: An Artificial Intelligence Approach*, Palo Alto: TIOGA Publishing Co., pp. 331-363, 1983a.

Michalski, R.S. and Stepp, R., “Automated Construction of Classifications: Conceptual Clustering versus Numerical Taxonomy,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 4, pp. 396-410, 1983b.

Poveda, P., “Classification of Folksongs According to Numerical Taxonomy,” unpublished report, University of Illinois at Urbana-Champaign, 1980.

Quinlan, J.R., *C4.5 Systems for Machine Learning*, Morgan Kaufmann Publishers Inc., 1993.

Seeman, W.D. and Michalski, R.S., “The CLUSTER3 System for Goal-oriented Conceptual Clustering: Method and Preliminary Results,” *Proceedings of The Data Mining and Information Engineering 2006 Conference*, Prague, Czech Republic, July, 2006.

Wojtusiak, J., “AQ21 User’s Guide,” *Reports of the Machine Learning and Inference Laboratory*,” MLI 04-3, George Mason University, September, 2004.

Wojtusiak, J. and Michalski, R. S., "The LEM3 System for Non-Darwinian Evolutionary Computation and Its Application to Complex Function Optimization," *Reports of the Machine Learning and Inference Laboratory*, MLI 05-2, George Mason University, Fairfax, VA, October, 2005.

Wojtusiak, J. and Michalski, R. S., "The LEM3 Implementation of Learnable Evolution Model and Its Testing on Complex Function Optimization Problems," *Proceedings of Genetic and Evolutionary Computation Conference*, GECCO 2006, Seattle, WA, July 8-12, 2006.

Wojtusiak, J., Michalski, R. S., Kaufman, K. and Pietrzykowski, J., "Multitype Pattern Discovery via AQ21: A Brief Description of the Method and Its Novel Features," *Reports of the Machine Learning and Inference Laboratory*, MLI 06-2, George Mason University, Fairfax, VA, 2006.

Yao, Y., Wang, F., Zheng, D. and Wang, J., “Rule + Exception Strategies for Security Information Analysis,” *IEEE Intelligent Systems* 20, pp. 52-57. 2004.

A publication of the *Machine Learning and Inference Laboratory*
College of Science
George Mason University
Fairfax, VA 22030-4444 U.S.A.
http://www.mli.gmu.edu

Editor: R. S. Michalski
Assistant Editor: K.A. Kaufman

The *Machine Learning and Inference (MLI) Laboratory Reports* are an official publication of the Machine Learning and Inference Laboratory, which has been published continuously since 1971 by R.S. Michalski's research group (until 1987, while the group was at the University of Illinois, they were called ISG (Intelligent Systems Group) Reports, or were part of the Department of Computer Science Reports).