Reports

Machine Learning and Inference Laboratory

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

College of Science

George Mason University

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, engineering, tax fraud detection, and musicology. Most of the results were obtained by applying our most 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 rule learning methods, such as C4.5, RIPPER, and CN2, as well as between conceptual clustering and conventional clustering, the last two sections describe results from applying these two classes of methods to two 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 done by our collaborator at Lockheed Martin Corporation, and the example of application to civil engineering, done in 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 of study. The amount of data collected may be very large, as in, for example, particle physics or tax return data, where it can be on the order of terabytes, or relatively small, as in archeology or criminology, where only a very limited amount of facts may be available.

Typical tools for data analysis and data mining have evolved primarily from research in statistics, and are now easily available in the form of commercially developed statistical software. New methods in this area may come from research on Computational Statistics, an area that is concerned with computationally-intensive methods of statistics and statistical visualization. Statistical tools for data analysis have been widely used, and can be very useful, but they also have significant limitations. They do not work well when the amount of available data is small, and they are primarily oriented toward creating numerical characterizations of the studied phenomena.

In many situations, however, it is desirable to characterize observations or express hypotheses about them not quantitatively (numerically), but rather qualitatively (descriptively), with an accompanying statistical annotation. Such qualitative descriptions may not only be sufficient for decision making, but may also be more reliable. For example, for everyday decision making, it may be quite sufficient and preferable to learn that the next week will be sunny, warm, and with low humidity, as opposed to getting a set of numbers describing precisely the temperature and humidity during each day of the week. Making an accurate qualitative prediction is also much easier than making an accurate quantitative prediction.

The 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 forms similar to those in which people express knowledge, such as natural language-like descriptions, figures, pictures and other visualizations. Such forms are easy to understand, interpret, and use for creating mental models. Basic methods of knowledge mining conduct what can be called *qualitative data analysis* that seeks human-style explanations of data, determines task-relevant information in large volumes of data with many irrelevant facts, hypothesizes logical implications from given facts, discovers structural relationships or causal dependencies, and incrementally improves previously determined qualitative knowledge in the light of new data.

This article reviews a range of practical applications of two methodologies of knowledge mining developed in our laboratory, *natural induction* and *conceptual clustering*. It is written in a tutorial fashion in order to make is easy to read by a wide range of readers. Sections 2-10, describe selected applications of natural induction, Sections 11-13 describe applications of conceptual clustering, and Sections 14-15 present examples of comparative studies with 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 and discovers patterns in data, expressing them in qualitative forms easy for people to understand and interpret, such as natural language-like descriptions and concept association graphs and generalized logic diagrams. Natural induction puts equal emphasis on predictive accuracy and the understandability of computer-generated knowledge, in contrast to standard methods, for which high predictive accuracy is the main or only objective.

Natural induction 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), which 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.

The hypotheses (rulesets) generated by AQ21 can be either generalized complete and consistent data descriptions, or strong patterns that can be partially inconsistent or incomplete. 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):

$$\text{LEF} = \langle \text{QUALITY}, q\%; \text{SIMPLICITY} \rangle$$

where:

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

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

In this measure:

Cov = p/P ("Coverage")

Consig = ((p / (p + n)) – (P /(P + N))) * ((P +N) /N) ("Consistency 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 Consig 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 operator costs were as follows: conjunction – 4, disjunction - 10, internal disjunction – 2, range – 2, less than or greater than operators – 1, equality operator – 1, inequality operator – 2. For operators

[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 for simplicity 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). 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 many criteria.

within an exception, 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 one 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.

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 an a priori defined mathematical measure of similarity. Conceptual clustering outputs both clusters and cluster descriptions, whereas conventional clustering outputs only clusters. 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 such an analysis of a medical dataset describing gene arrays 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.

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 using gene arrays (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 array 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 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 gene array 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.

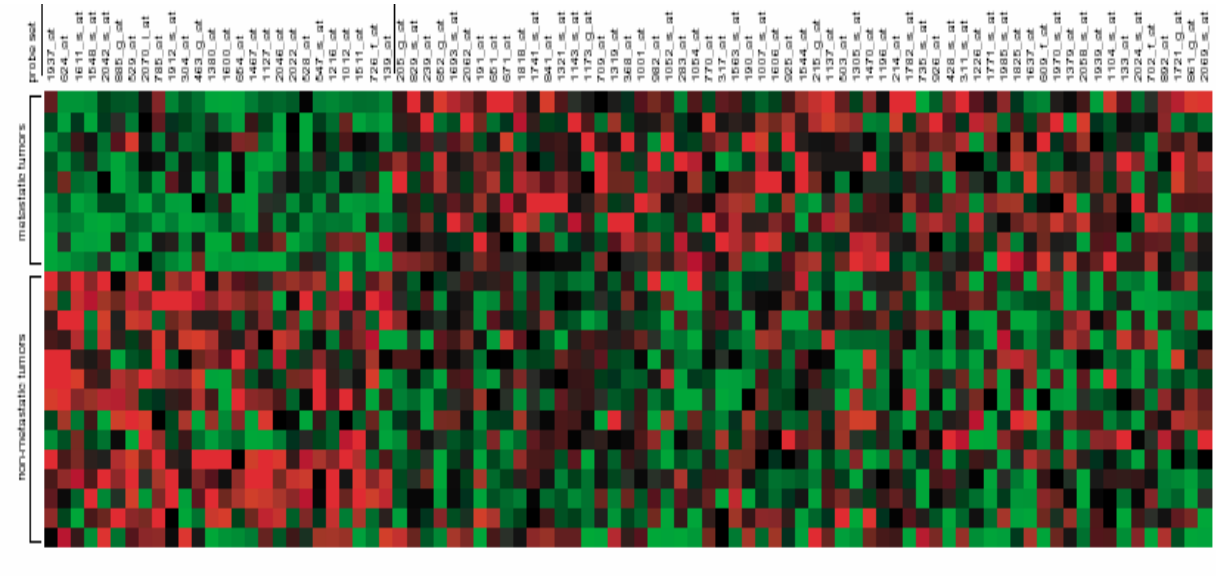


Figure 1: A gene array containing medulloblastoma data for 23 patients.

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 attributes, we applied AQ21 natural induction program to hypothesize rules for distinguishing between metastatic and non-metastatic rumors.

In the experiment reported below, 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 for the metastatic tumor presented in Figure 2.

In Figure 2, the condition [Cancer = metastatic] is the rule *consequent* that is implied by two alternative premises (that follow the implication sign "<="). A pair, a consequent and a

premise, constitute 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.

[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.

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, "p_c"), the second number indicates the number of negative examples (here, non-metastatic patients that satisfy this condition (denoted, "n_c"), and the third number expresses 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 premise of the rule. Thus, in the first condition of each rule, both triples are always identical, because there are no previous conditions. 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 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).

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.

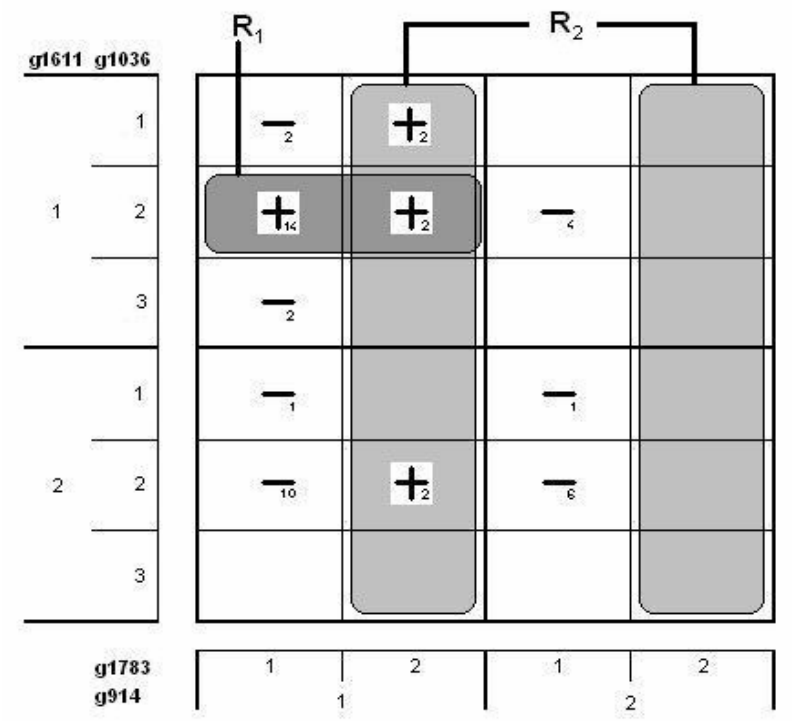


Figure 3: GLD visualization of rules for detecting metastatic tumors.

To visualize more complex rulesets, 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 rules. The thicknesses of the links represent a requested measure of the condition importances. Depending on the parameter setting, the program can use different importance measures, such as *condition consistency* (p_c / (p_c + n_c)) or *condition support* (p_c).

For example, Figure 4 presents a concept association graph visualizing the discovered rules for medulloblastoma. In this graph, each link is annotated by a pair of numbers (p_c, n_c), and its thickness is proportional to the condition consistency.

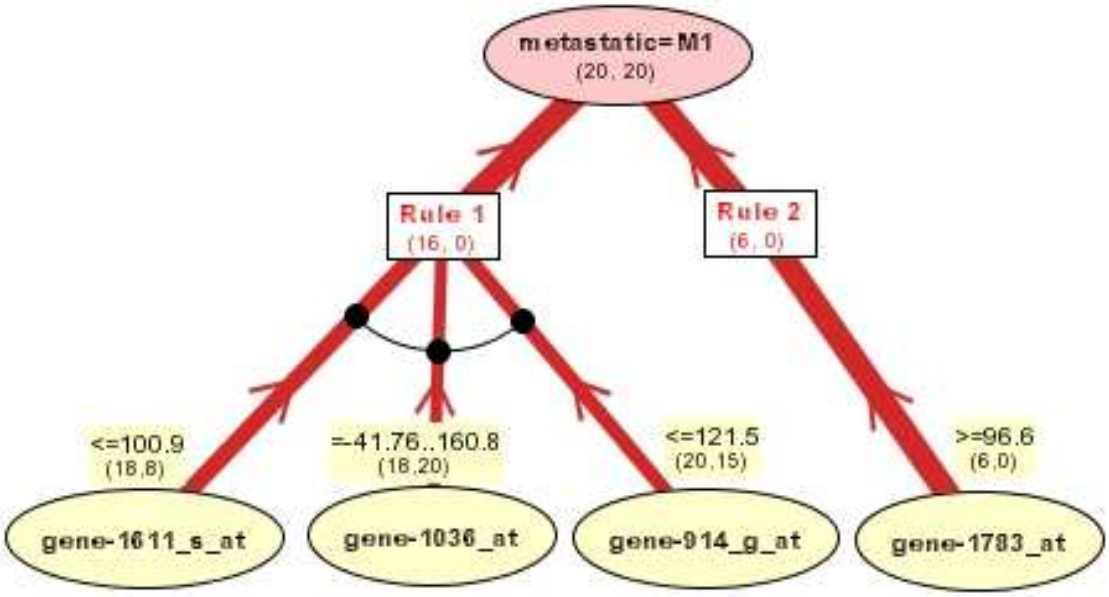


Figure 4: A CAG visualizing discovered rules for recognizing metastatic tumors.

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 the over 2000 genes. When the experiment was performed using 5-fold cross-validation[2], the predictive accuracy was about 95%.

The ruleset is surprisingly simple and easy to understand. Despite having a very limited number of training examples to learn from, the natural induction method also achieved high predictive accuracy. These results from natural induction are in contrast with the results from the neural net developed by MacDonald et al. for the same task, which requires measuring 80 genes (attributes), and whose reported predictive accuracy is about 72%. Moreover, the neural net presents a "black box" solution that is difficult to interpret and understand, while the "transparent box" ruleset obtained by natural induction can be easily interpreted.

Further research is needed to test the rules obtained by natural induction on more data and to have them evaluated by medulloblastoma experts before they could be accepted as a valid medical discovery.

---

[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.

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 denoting 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 25 diseases and the lifestyles and other diseases. Among the many discovered patterns, a typical example is shown in Figure 5.

[Arthritis = Present]
<= [HBP=present: 432, 1765] &
[Education<=college_grad: 940, 4529] &
[Rotundity>=low: 1070, 5578] &
[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, no education beyond college, more than very low rotundity, 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)). The most significant factor in the pattern is high blood pressure, which by itself has consistency of about 19%.(p_c / p_c +n_c).

Discovered patterns were visualized using concept association graphs. One such graph is presented in Figure 6. In this graph, a link's thickness reflects the condition consistency, 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.

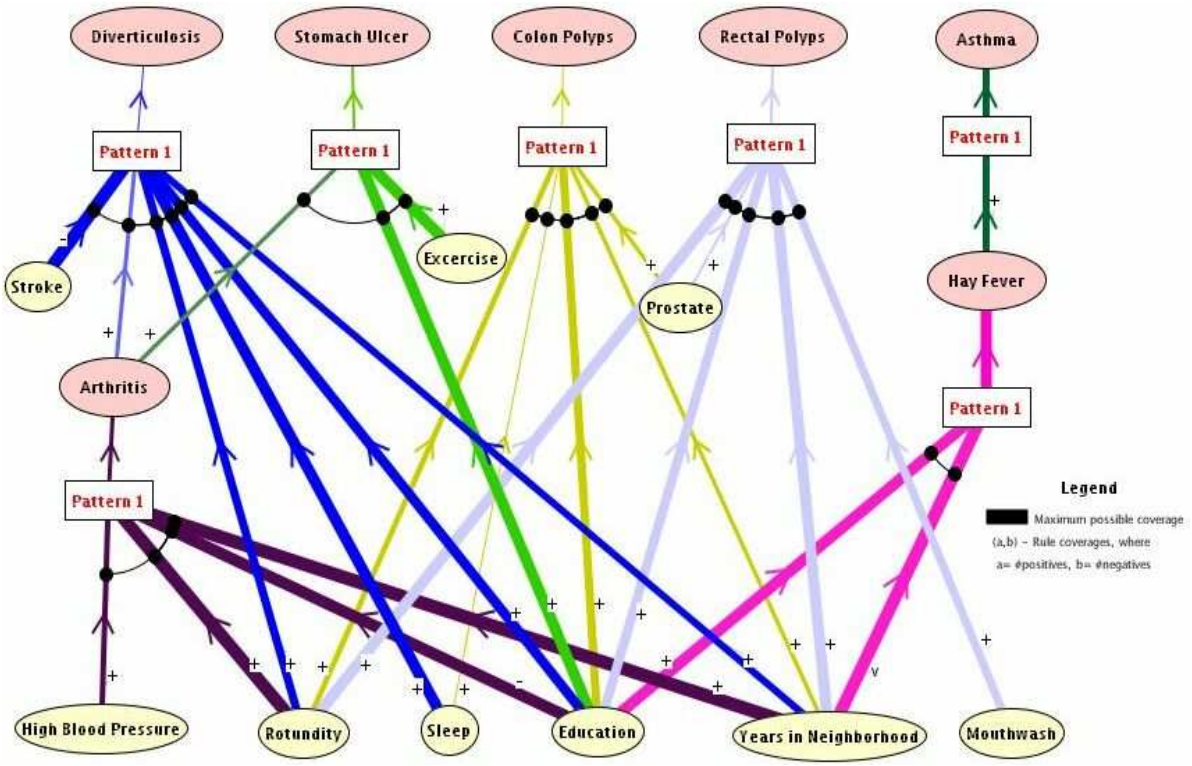


Figure 6: Concept Association Graph representing seven patterns in the medical database.

The graph in Figure 6 was automatically generated from the discovered patterns. 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 an application of natural induction that generates *attributional rule-tree* representations, 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 rules 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 flat ruleset learned for the fifteen soybean diseases.

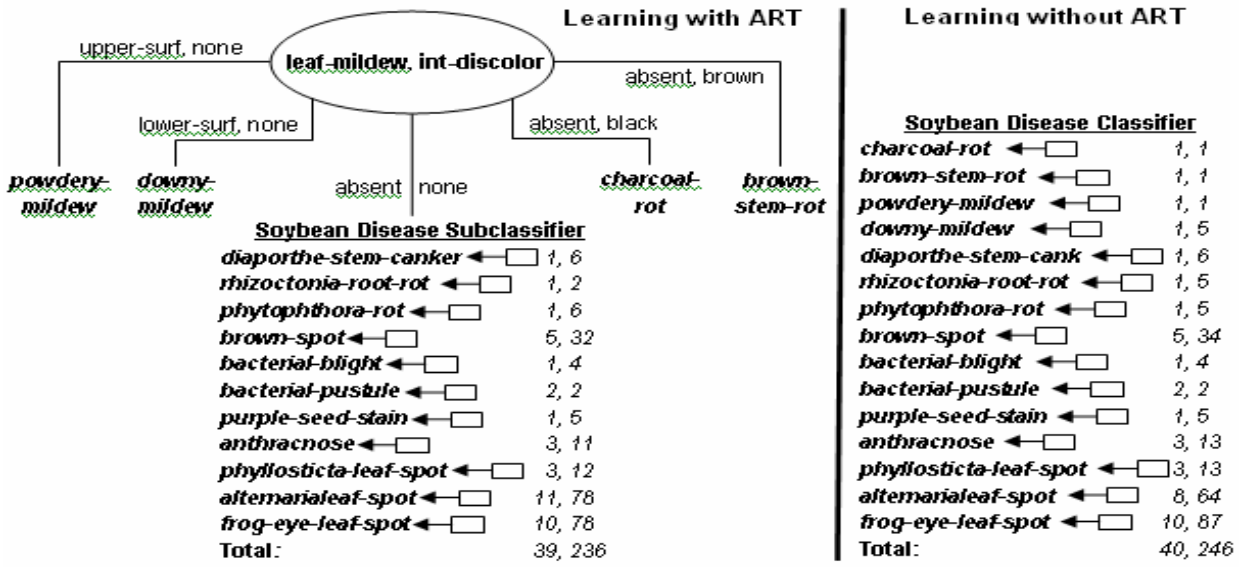


Figure 7: Complexity comparison of rule trees and rulesets in the soybean domain.

The 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 to 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 nineteen 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: Selected attributes 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 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=228, 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, or 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 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 9171 others. The second condition sets values on the process name at all four positions of the multigram, and other conditions are interpreted similarly. 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 in question (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 we measured a sufficient 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 is 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.

Although these experiments have explored only a small subspace of possible experiments on learning and testing of the developed user modeling methods, they show that LUS-MT 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 sufficiently similar to that recorded in the training datastream.

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

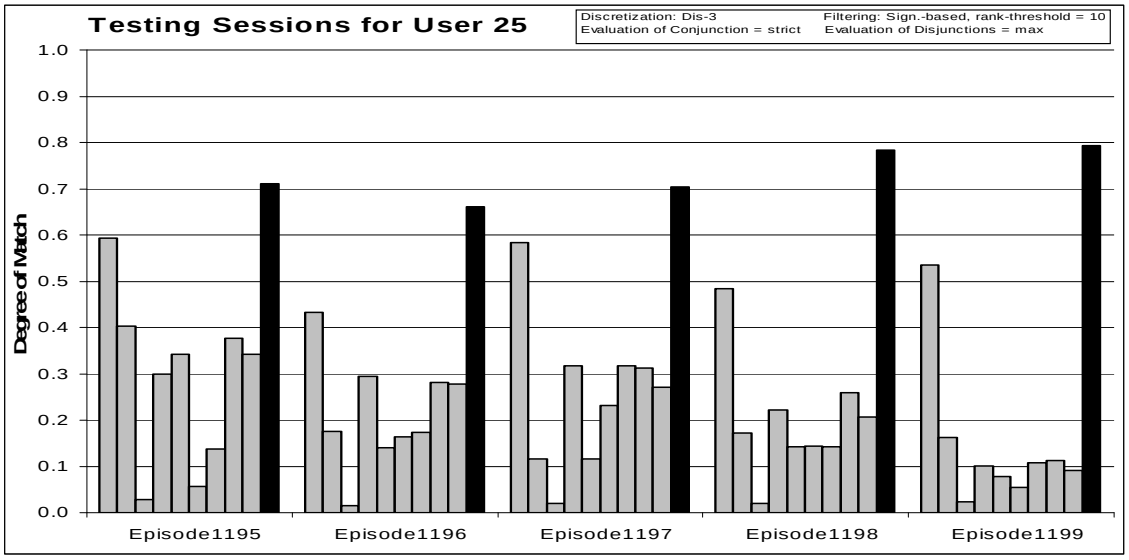


Figure 11: Classification of User25 test episodes.

8 AN EXAMPLE OF APPLICATION TO COMPLEX SYSTEM OPTIMIZATION

This section describes an application of natural induction to guiding evolutionary optimization of complex systems. The specific example presented here concerns 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 12). 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), invented in our laboratory.

LEM is a form of non-Darwinian 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.

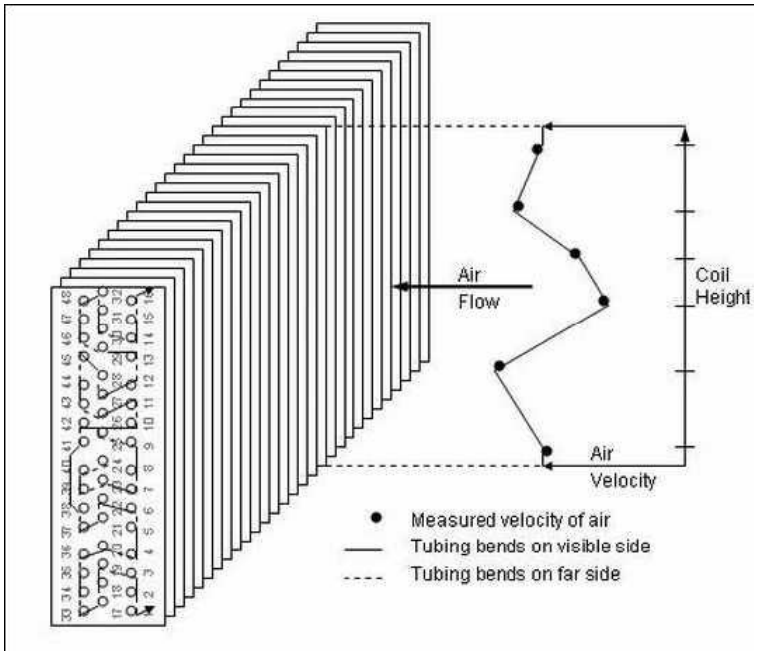


Figure 12: An example of a heat exchanger.

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.

9 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. Such specifications may include the size, mount, motor, flange, variable speed drive, gear ratio, and model number. Given a specification, the manufacturer must either look up the components of that gearbox or, if no gearbox with those specifications has previously been requested, determine the components needed to build the requested gearbox (e.g., which shaft, which flange, which lubricant). The hope was that an inductively learned rule base would provide assistance in this regard. This work was conducted in collaboration with Lenze GmbH & Co KG, Germany.

Figure 12 shows exemplary rules for selecting the Schnekenwelle (worm shaft) determined by AQ learning system (AQ19). For instance, the first rule says to use part number 00650943 for the worm shaft when the gear ratio is 5 and the model number is 104, 105, 113 or 145. Other rules are interpreted similarly.

Some of the discovered rules, such as those in Figure 13, were straightforward. For the requested size or gear ratio, certain parts were dictated. Other rules were more complex. The learned rules provide insights into the relationships and constraints of the gearbox manufacturing domain, and even exposed some errors in the data that had been provided.

[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 13: Examples of discovered rules for assigning the correct *Schnekenwelle* (worm shaft) the given the gear ratio, model number, and motor size.

The learned classifier 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 consistent and complete with all the data. This result was quite satisfactory for the Lenze company, not only because of the high accuracy of the rules, but also because they were so easy to interpret.

10 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 depend on the additives used in it. In the application 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: “Class” with the domain {Cement Paste, Interface region, Aggregate, Void}, and nine numeric attributes representing the number of cases measured (Lzd), average energies (Sen), and average amplitudes (Saz) in three energy bands (H, M, L). A decision attribute, “Composition,” had the three-valued domain {silica,

no additives, pfa}. The training dataset consisted of 500 examples, with 302 examples in which silica was 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 author 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).

11 AN EXAMPLE OF APPLICATION OF CONCEPTUAL CLUSTERING TO AGRICULTURE

An important research question in the area of unsupervised learning is how well concepts discovered by a machine learning program correspond to categorizations developed by experts. This problem was investigated by applying conceptual clustering program CLUSTER/2 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 re-create a classification that would cluster based on which of four selected diseases, Diaporthe stem canker (denoted D1), Charcoal rot (D2), Rhizoctonia root rot (D3), or Phytophtora rot (D4), was present. There were 47 cases selected, initially described by 35 mutlivalued 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.

The description of the cluster generated by CLUSTER/2 that corresponds to D1 is shown in Figure 14(a). Figure 14(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 Figures14 (a) and (b), the program-generated description for cluster D1 corresponds very 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 reflected the input data.

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 quite limited dataset, but also created a description of diseases that matches well their expert-provided characterization. For more details on this research and on conceptual clustering, consult (Michalski and Stepp, 1983a,b; Seeman and Michalski, 2006).

[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) CLUSTER/2-generated description	(b) Expert-provided description

Figure 14: Two descriptions of Diaporthe stem canker in soybeans.

12 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), consisted of descriptions of 100 Spanish songs. Each song was described in terms of 22 attributes, some were binary, some discrete and some continuous. The clustering evaluation criterion was to seek clusters whose description had minimum sparseness, (that is, that covered the minimum number of unobserved cases while covering the maximum number of observed cases). The taxonomy automatically generated by the conceptual clustering program is shown in Figure 15.

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 conjunctive description of classes of songs associated with different leaves.

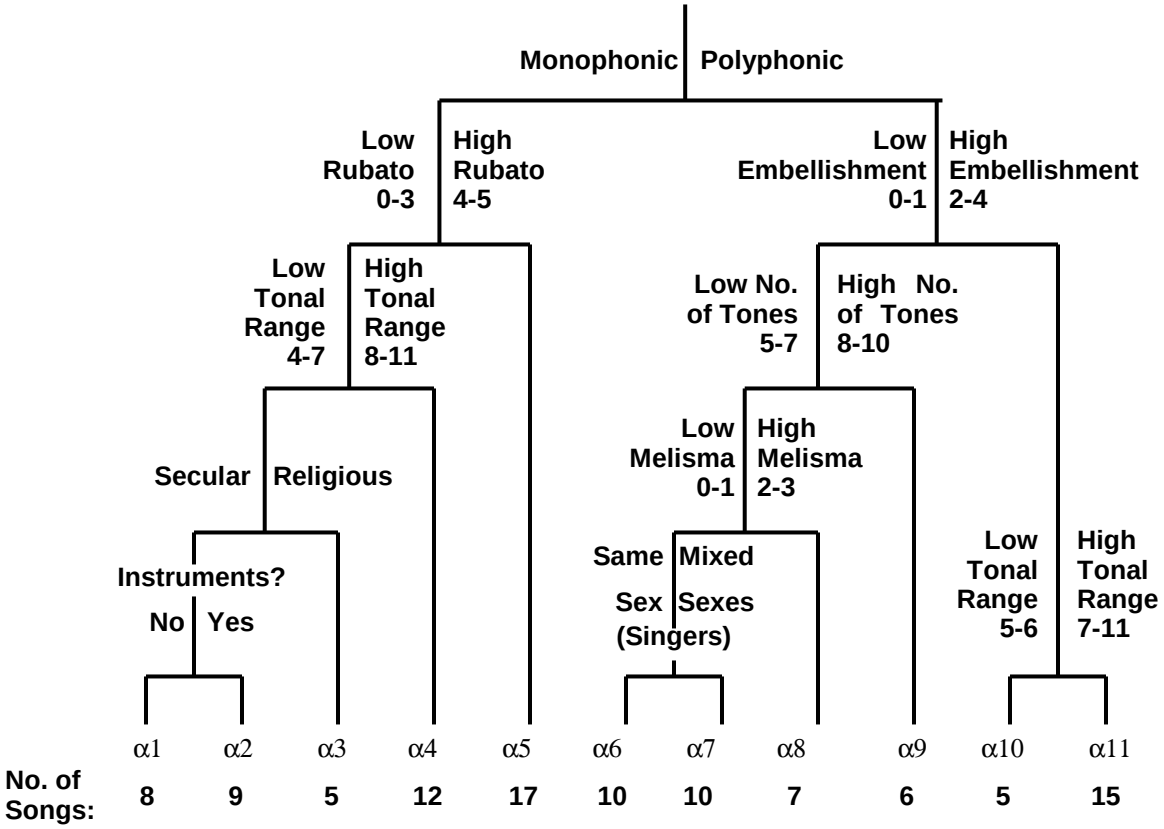


Figure 15: A classification hierarchy of Spanish folk songs produced by conceptual clustering (figure reproduced from Michalski and Stepp, 1983a).

This result was positively evaluated by the musicologist, because it corresponded 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.

13 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 16 depicts this methodology.

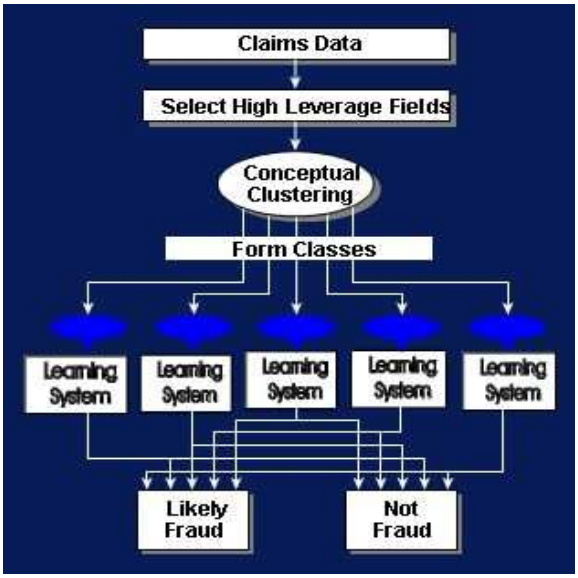


Figure 16: Clustering and rule learning for tax fraud detection (the diagram made by Scott Fischthal).

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 17).

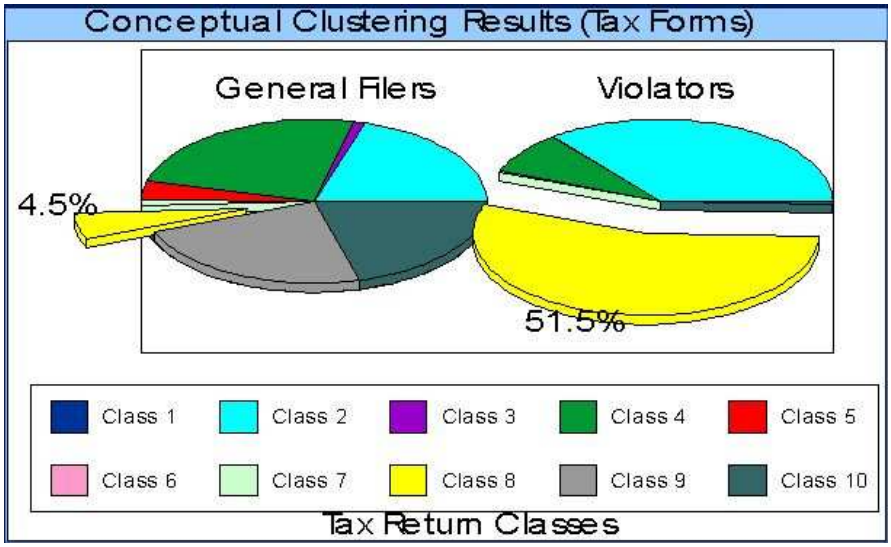


Figure 17: Distribution of filers and cases of fraud among the discovered groups (result by Scott Fischthal).

The above means that taxpayers with the profile satisfying the description of the discovered Class 8 cluster are much more likely to submit a fraudulent tax form return than those that do not satisfy that description.

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

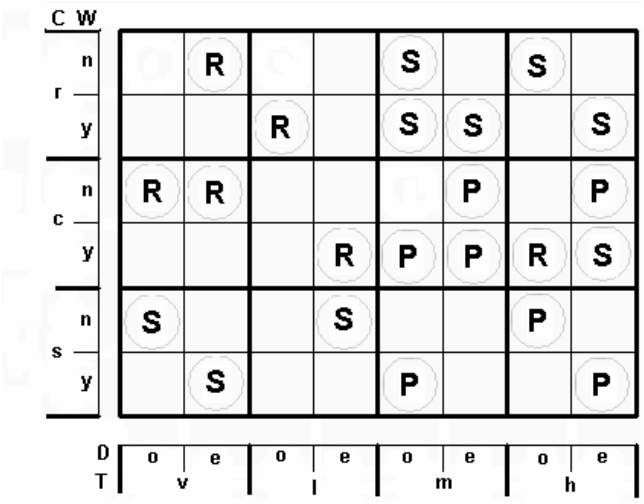
14.1 Problem Definition

In order to help the reader get an insight into the natural induction capabilities of the AQ21 program and those of some other well-known programs, we designed a simple, easy to understand problem. In this problem, given entities are described in terms of the attributes presented in Figure 18.

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 18: 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 19 presents a General Logic Diagram spanned over the input attributes. 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 has not been determined.



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

14.2 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 20 presents a decision tree and Figure 21 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 given event 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 20: A decision tree determined 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 21: Decision rules derived by C4.5.

The RIPPER program applied to the same dataset determined the rules presented in Figure 22. These rules need to be evaluated sequentially; thus 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 22: Rules learned by RIPPER.

Decision rules determined by the CN2 program are presented in Figure *23*.

```
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 23: Rules determined by CN2.

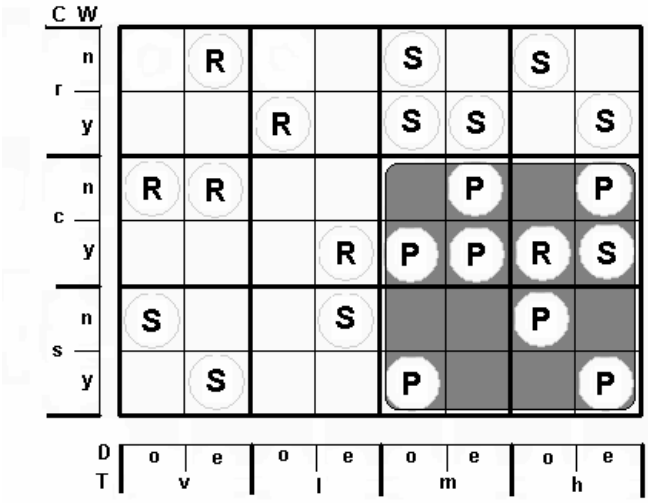
The subsequent Figures 24, 25, 26, and 27 present different rules determined by AQ21. Section 14.4 summarizes the predictive accuracy of the descriptions generated by different programs for the training and testing data sets. In these experiments, the testing set consisted of all events that are represented by empty cells in Figure 19. The predictive accuracy on the testing set thus evaluates the quality of generalization performed by the different programs.

The presented results were obtained by the AQ21 program using different settings of the program parameters. With the default parameter setting that instructed the program to determine a strong pattern for the Activity "Play," AQ21 determined the pattern presented in Figure 24.

The pattern consists of one 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* had the default value 0.5 (requesting an equal emphasis on completeness and consistency). Numbers inside conditions represent positive and negative coverages of the conditions on their own. The pattern is graphically illustrated in Figure 25.

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

Figure 24: A strong pattern discovered by AQ21.



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 25: GLD showing the strong pattern discovered by AQ21.

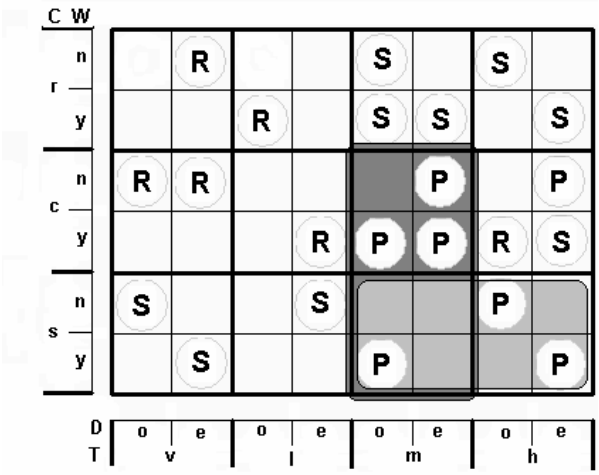
AQ21 allows the user to control the tradeoff between completeness and consistency of patterns by adjusting parameter *w* in the pattern quality measure q(*w*) (Michalski and Kaufman, 2001). When setting the *w* parameter to 0.15, AQ21 found two rules presented in Figure 26 and graphically in Figure 27. The pattern is consistent but almost complete – one training example is not covered by the learned rules.

Note that further reduction of *w* to 0 leads to a complete and consistent ruleset, which can be obtained directly by executing AQ21 in Theory Formation mode.

```
[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 26: Rules found 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 27: GLD showing the pattern found by AQ21 for *w* = .15.

14.3 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. The processes of learning 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 produced the rule with EXCEPTION presented in Figure 28. 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.

```
[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 28: Strong pattern with exception found by AQ21.

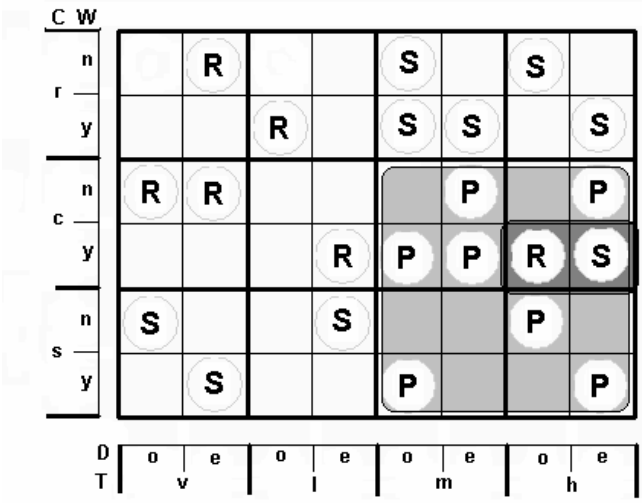
Figure 29 shows a GLD representation of the rule presented in Figure 28. The two highlighted examples representing "shop" (S) and "read" (R) activities are treated as exceptions to the general pattern for the play activity.

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.



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

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

14.4 Discussion of Results

The obtained results are summarized in Table 2. The C4.5-generated decision tree has 3 internal nodes and *11* branches, and when applied to the training dataset (22 examples) gave

4 errors (18%). The C4.5 rules (7 rules) gave 5 errors (23%). RIPPER produced 6 rules that gave 4 errors (18%) on the training dataset. CN2 produced 13 rules the constituted a complete and consistent description, but comprised by far the most complex description.

In Pattern Discovery mode using the default value of w (0.5), AQ21 produced one strong pattern that covered all positive examples and 2 examples of other classes (9% error). When executed with w=0.15, it produced a consistent description and nearly-complete, as it missed one example of activity play (4.5% error). In Theory Formation mode, AQ21 generated three simple rules which were complete and consistent with the training dataset (0% errors). AQ21 with exceptions found one strong pattern that was 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.

In this simple example, AQ21 produced both more accurate and simpler patterns than the other programs. This result was achieved due to the richer representation language used by this program. AQ21's more expressive representation language facilitates generation of simpler hypotheses and in the forms that are easy to interpret in natural language. AQ21 also allows the user to control the type of output by parameters and the hypothesis quality criterion. The latter feature is useful when different problems may require different criteria of pattern optimality.

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)

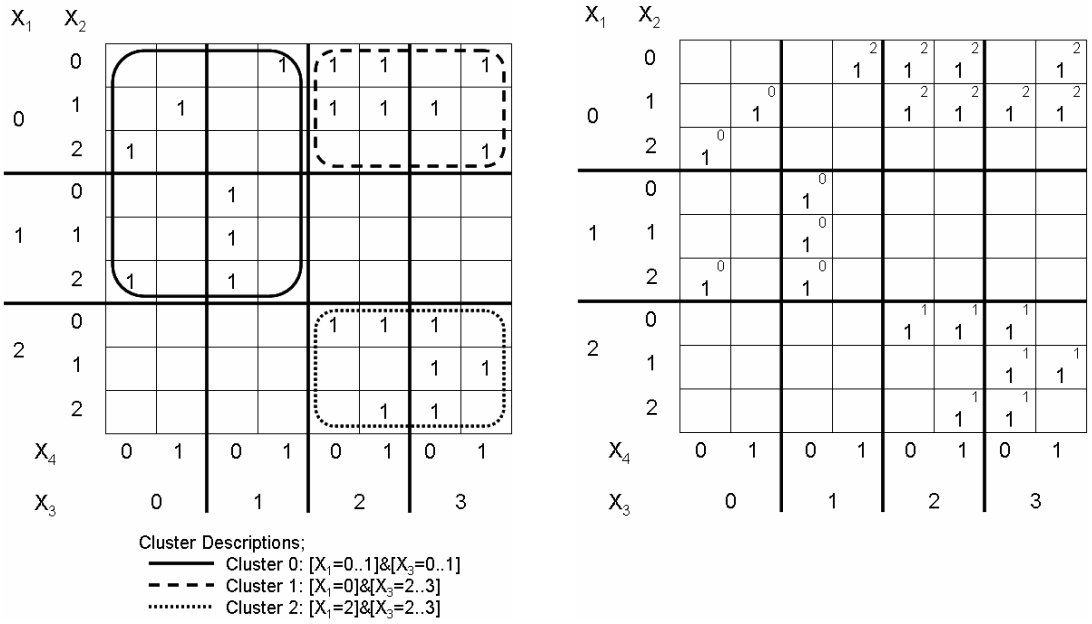
15 COMPARING CONCEPTUAL CLUSTERING WITH CONVENTIONAL CLUSTERING ON A SIMPLE DESIGN PROBLEM

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

The objects in the dataset to be clustered in this problem are described by four attributes with domains as follows: X1: {0,1,2}, X2: {0,1,2}, X3: {0,1,2,3}, and X4: {0,1}. The dataset consists of 21 tuples distributed in the space spanned over these four attributes.

The dataset was tested against the implementation of Lloyd's algorithm in the KMlocal clustering application (Kanugo et al., 2002). Lloyd's algorithm assigns observations to clusters using the minimum Euclidean distance between the observation and the cluster centroids. The KMlocal application was run with default parameters and *1000* stages. The CLUSTER3 conceptual clustering application was run with default parameters. The number of clusters was set to 3 for both applications.

The three simple disjoint cluster descriptions produced by CLUSTER3 are listed and represented visually by the GLD in Figure 30(a). A similar GLD in Figure 30(b) indicates the results of the KMlocal application. In Figure 30(a), ellipses indicate the conceptual descriptions of the clusters; no ellipses are shown in Figure 30(b) since the results are groups of datapoints and not generalized descriptions. Instead, the cluster number is indicated in the upper right-hand corner of the observation cell.



(a) Clusters generated by CLUSTER3 (b) Clusters generated by KMlocal

Figure 30: A GLD representation of clusters of the designed dataset.

The clusters produced by KMlocal and CLUSTER3 are similar, except for the observation (0,0,1,1). The value of X_2 for this point is a contributing factor for its inclusion by KMlocal in cluster 2, rather than cluster 0. However, in addition to clusters, CLUSTER3 produced their description in the form of atttrbutional conjunctions, and this observation fell naturally into the cluster defined by $[X_1 = 0..1]$ & $[X_3 = 0..1]$, rather than the one defined by $[X_1 = 0..1]$ & $[X_3 = 2..3]$.

The descriptions produced by CLUSTER3 are general, in the sense that they not only cover the observations in the dataset, but also unobserved instances. Thus, news observations can be easily classified by determining which description it matches. For example, the observation (0,2,1,1) is classified to Cluster 0 because it satisfies the description of that cluster. In contrast, in the case of KMlocal, it is not obvious whether this observation belongs to Cluster 0 or Cluster 2. To decide this, the system would need to re-execute the KMlocal application upon the entire dataset.

16 CONCLUSION

This paper briefly described and provided examples of applying natural induction and conceptual clustering to a wide range of real world problems. To give some insight into the differences between the natural induction method and several other well-known methods, it also presented results of comparing our method with the other methods, using a simple, designed problem. The final section also provides a very simple illustration of the differences between conceptual clustering and conventional similarity-based clustering. To get a more adequate insight into the differences between natural induction, conceptual clustering and other methods developed for similar tasks requires their testing on much more complex problems.

For technical details on the methods of natural induction and conceptual clustering employed in the applications described here, and for the downloadable program AQ21, visit http://www.mli.gmu.edu. This site also provides many papers on these topics. 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 showing the applicability of the described methods to a wide range of problems, we hope to possibly interest readers 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.

Kufman, 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 (to appear).

Wojtusiak, J., "AQ*21* User's Guide," *Reports of the Machine Learning and Inference Laboratory*," MLI 04-3, George Mason University, September, 2004.

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).