Communications of the Association for Information Systems

Volume 5 Article 6

January 2001

GPR: A Data Mining Tool Using Genetic Programming

Balasubramaniam Ramesh
Georgia State University, bramesh@gsu.edu

Communications of the Association for Information Systems

Volume 5, Article 6
April 2001

GPR: A Data Mining Tool Using Genetic Programming

Balasubramaniam Ramesh
Department of Computer Information Systems
Georgia State University

Tung Bui
Department of Information Technology Management
University of Hawaii

bramesh@gsu.edu

RESEARCH

GPR: A DATA MINING TOOL USING GENETIC PROGRAMMING

Balasubramaniam Ramesh
Department of Computer Information Systems
Georgia State University

Tung Bui
Department of Information Technology Management
University of Hawaii

bramesh@gsu.edu

ABSTRACT

This paper proposes an inductive data mining technique (named GPR) based on genetic programming. Unlike other mining systems, the particularity of our technique is its ability to discover business rules that satisfy multiple (and possibly conflicting) decision or search criteria simultaneously. We present a step-by-step method to implement GPR, and introduce a prototype that generates production rules from real life data. We also report in this article on the use of GPR in an organization that seeks to understand how its employees make decisions in a “voluntary separation" program. Using a personnel database of 12,787 employees with 35 descriptive variables, our technique is able to discover employees’ hidden decision making patterns in the form of production rules. As our approach does not require any domain specific knowledge, it can be used without any major modification in different domains.

Keywords: data mining, genetic programming, genetic algorithms, rule induction, knowledge discovery, human resource management

I. INTRODUCTION

Most large organizations possess tremendous amounts of data stored in databases including financial information, personnel records, manufacturing data inventory information, and customer information. These data are accessed to produce reports, statistics, and business queries. Corporate managers finding themselves in the possession of large and rapidly growing databases are beginning to suspect that, despite the large amount of available output, information in their databases is not used to the fullest potential. With the limitations imposed by cognitive capabilities, they are unlikely to discover any but the most obvious and uninteresting patterns in the massive data. Mechanisms to find underlying patterns of behavior hidden in databases in critical business areas such as market intelligence, manufacturing process control, purchasing, and inventory management, can provide invaluable competitive advantage to the organization that uses them [Chung and Gray, 1999, Dhar, 1998]. The use of automated systems to find new knowledge is necessary and worthwhile because it is neither feasible nor cost effective to examine, analyze, and interpret the typically large corporate database manually in this pursuit [Smyth and Goodman, 1992].

In this paper, we present a novel approach to data mining that uses the principles of genetic programming to generate production rules from databases. Our approach is unique in that it easily accommodates knowledge discovery satisfying any user-specified criteria and is generic enough to offer wide applicability in a large number of data mining applications.

We present GPR, an inductive data-mining system we developed. GPR uses the technique of genetic programming to discover rules. In the following section, we briefly define terminology and concepts related to knowledge discovery and the reasons for our focus on discovering production rules. In Section III we discuss the application of genetic programming to data mining.

Section IV provides a detailed description of GPR, our prototype data mining tool. We illustrate the use of genetic programming for data mining with a detailed case study in Section V of a real-life application in military manpower management. In this Section we also present and discuss the significance of the results from GPR. The last two sections discuss related work and presents conclusions.

II. DISCOVERY OF PRODUCTION RULES

Knowledge discovery was defined by [Frawley et al., 1991] as "the non-trivial extraction of implicit, previously unknown, and potentially useful information from data". Zytkow [1993] defines *knowledge discovery* as the "acquisition of objective knowledge" as distinct from *learning,* which he defines as acquiring knowledge that is already known.

Two basic processes are used to infer knowledge from raw data: deduction and induction.

- *Deduction* is the process of reasoning from the general to the specific. It is a learning process that involves drawing specific conclusions rationally from more general principles, which are assumed to be true. Deduction allows inference of specific knowledge about relationships between data elements in the database, much as a syllogism is constructed. Systems derived from human expertise typically use deductive reasoning.

- *Induction* is the process of reasoning from the specific to the general. As a discovery process, it involves drawing conclusions based on generalized patterns found in the facts. Data mining applies inductive reasoning to databases containing facts. Each pattern discovered using data mining is a piece of knowledge. The collection of discovered patterns constitutes a model of the database.

Thus, inductive systems discover the information that no one knows to ask for, whereas deductive systems provide data to support the patterns that a user wishes to analyze.

DATA MINING CLASSES OF PROBLEMS

Data mining often uses techniques developed in the field of machine learning to extract valuable high-level information from databases. It attempts to find the patterns with the greatest utility to the user. Classification, sequencing and association, are among the primary classes of problems addressed by most data mining systems.

- *Classification* involves partitioning observations in the database into groups. Examples of applications of classification systems include credit approval and determining appropriate treatment for patients.
- *Sequencing* involves finding connections among (temporally) ordered data [Agrawal et al., 1993]. Typical applications of sequencing include modeling stock market movements and weather forecasting. A simple example is purchase of a house is followed by purchase of appliances such as refrigerator and microwave.
- The problem of *association* involves generalizing patterns discovered in the database. In the association mode, data mining attempts to discover and describe knowledge about a specified (target) database field(s) or attribute(s) in terms of other (non-target) attributes. Associations are commonly expressed as sets of rules satisfying some specification. A data mining system, for example, might attempt to discover regularities or rules, by analyzing a number of instances or examples of data related to a problem. An example of this category is a rule about detection of faults in a manufacturing process.

A class, together with its description, constitutes a classification rule:

"If <attribute description> then <class>".

An associative pattern can be used to predict the value of new examples or to understand or explain the data from which the pattern was derived.

KNOWLEDGE REPRESENTATION

The intended use of the discovered knowledge should guide the way the results of the data mining system are represented. Discovered knowledge is frequently represented in associative data mining systems by one of two ways: *decision trees* or *production rules*. Other representational formats such as neural networks, semantic nets, and decision lists are not discussed here since there are less frequently used in applications requiring understandability by human decision-makers. The representation often drives the logic by which the knowledge is derived. Regardless of the representation format, it is important to remember that only descriptions of relationships are expressed. The conditions necessary to support causation may not be present.

DECISION TREES

A decision tree represents knowledge in the form of a map or tree of the relations found among the data.

- Nodes of decision trees are labeled with attribute names;
- Edges are labeled with possible values for this attribute, and
- Leaves are labeled with the different classes of the target attribute.

Trees must always begin with the attribute associated with the root node and partition the data into branches based on values of attributes. An object is classified by following a path down the tree, along the edges corresponding to the value of the attributes of that object [Holsheimer and Siebes, 1994].

Systems generating decision trees use essentially sequential decision algorithms. Decision trees "tend to grow very large for realistic applications and are thus difficult to interpret by humans" [Holsheimer and Siebes, 1994]. Decision trees also grow excessively complicated in the presence of noisy databases [Dhar and Tuzhilin, 1993]. Most systems that use decision-tree representations use a pruning mechanism to offset this tendency to overfit noisy data [Quinlan, 1986]. Decision trees may be appropriate if the reasoning process is complex. While conjunctions are represented effectively, disjunctions require duplication and therefore, require large trees. Also, when attributes contain a large number of values, it is necessary to create subsets of attribute values to effectively reduce the amount of fragmentation necessary to represent the data accurately. Decision trees are particularly useful when the results of the data mining system will be input directly into other computer programs, but may be too complex for interpretation by humans [Frawley et al., 1991].

PRODUCTION RULES

Production rules represent relationships between attributes. Production rules used by data mining systems appear in the form:

"If *description y* Then *target attribute class x*"

where *y* is in terms of the non-target attributes.[1] A degree of certainty or confidence (the probability of *x* given *y*) is usually associated with production rules. The advantage of production rules is that they are familiar and are easily understood by humans [Holsheimer and Siebes, 1994]. For example, expert systems frequently represent knowledge in the form of production rules. Data mining systems generate rules that can be understood and used as machine-generated expertise. Because of their inherent clarity, production rules are most appropriate in decision support systems where human understanding of the underlying relationships between the attributes is necessary to take appropriate action.

In general, systems that generate rules are more flexible than systems using decision tree structures and more readily accommodate missing attribute

[1] This notation convention is the reverse of the traditional: If *x* then *y*.

information [Smyth and Goodman, 1992]. Quinlan [1988] summarizes several advantages of production rules, including:

- They are widely used in machine learning applications
- They are easily understood by both experts and users due to their simple structure
- Each rule can be interpreted and understood without reference to any other rules, and
- The classification accuracy of a decision tree can be improved by transforming the tree into production rules, thereby eliminating tests that are attributable to peculiarities in the training data[2]. For the above reasons, we use production rules as the representation of discovered knowledge in GPR.

III. USING GENETIC PROGRAMMING FOR DATA MINING

The primary task of a data-mining system is to search for general patterns that describe the classes of the designated attribute in terms of the other attributes. Evaluating every existing description in the database is the only way to ensure the best set of patterns is found. The disadvantage of this strategy is that it generates too many patterns, many of which are obvious, redundant, or useless [Piatetsky-Shapiro, 1999, Piatetsky-Shapiro et al., 1993]. Further, an exhaustive search is quite slow. The choice of the strategy depends on the needs for speed and accuracy for a given data mining application. For example, in a real-time production process control situation, a quick response is more important than finding the optimal solution. On the other hand, if the objective of the data mining system is to support back office productivity, (e.g., model the behavior of workers in order to design an incentive system to encourage productivity), time is less critical than the optimality of the solution. To

[2] Typically, the data used in building a model (for example, production rules) is divided into three sets. Training data set (also referred to as *in sample* data set) is the data set used to build a model. Test data set is the data set used to refine the model to prevent the model from memorizing the training data so that the model is more general and will work with unseen data.

accommodate different goals of a data-mining task, data mining search strategies are often guided by criteria called the *knowledge quality functions,* discussed in Section IV. Several search techniques developed in the field of Artificial Intelligence (for a survey, see Nilsson [1988]) find high-quality descriptions of the target attribute classes without exhaustively searching the space of all possible descriptions. This problem involves the search for inductive knowledge. The search for the set of high-quality descriptions can be viewed as an "optimization" problem. Genetic algorithms and genetic programming are effective in a variety of problem domains involving such search [De-Jong, 1999, Jorng-Tzong and Ching-Chang, 2000, Michalewicz, 1996, Wang, 2000]. The objective is to discover the "best" generalizations or hypotheses that satisfy certain conditions and best explain the data. Working hypotheses are generated, verified, rejected and improved upon until the requirements are met.

GENETIC ALGORITHMS: BIOLOGICAL PRINCIPLES AND APPLICATIONS TO DATA MINING

Introduced by [Holland, 1975], genetic algorithms (GA) imitate the process of biological natural evolution. In nature, individuals in a population compete with each other for resources. The genes of the most adapted and "fit" individuals are passed along to more individuals in the succeeding generation. The population adapts to the environment through a process of selection, reproduction, recombination, and mutation. Many of these processes favor individuals with higher "fitness" so that during evolution through several generations, the average quality of the population increases towards its own particular optimum. Genetic algorithms use operations analogous to these processes to propagate modifications of descriptions across iterations (generations). These biological principles are the basis for genetic algorithms.

The performance of the model is estimated on a third data set called the validation data set, which is distinct from the training and test data sets.

GA implements a genetic model of computation by having arrays of bits or characters to represent biological chromosomes. In data mining, potential solutions in a genetic algorithm are represented as strings of ones and zeroes, where parts of the strings correspond with one attribute (i.e., field of the database). A large number of strings are generated at random and evaluated for fitness. The quality of the strings, as measured by a quality (fitness) function, determines which strings will participate in genetic operations such as crossover and mutation [De-Jong, 1999, Grefenstette, 1988].

GA typically employs the following cycle (see Figure 1):

- First, an initial population of user-defined size is populated (often by random generation).
- At each generation, the fitness of all the individuals in the population is evaluated using some form of computational function called the *fitness function*. It measures the ability of a solution to solve the problem at hand.

  Performing genetic operations such as selection, crossover, and mutation creates a new generation. Selection involves choosing some chromosomes in the current generation to survive in the next. A common form of selection is one where each chromosome's likelihood of being selected is proportional to its fitness. Crossover involves selecting two individuals from the current population as parents. Their chromosomes are exchanged randomly and two children are produced, each of which has some combination of its parents' genetic information. Mutation occurs by selecting a single parent and producing random changes in a small number of its chromosomes to produce a child.

  Once the new generation is produced, the old population is discarded and the process is repeated until the termination criteria are met.

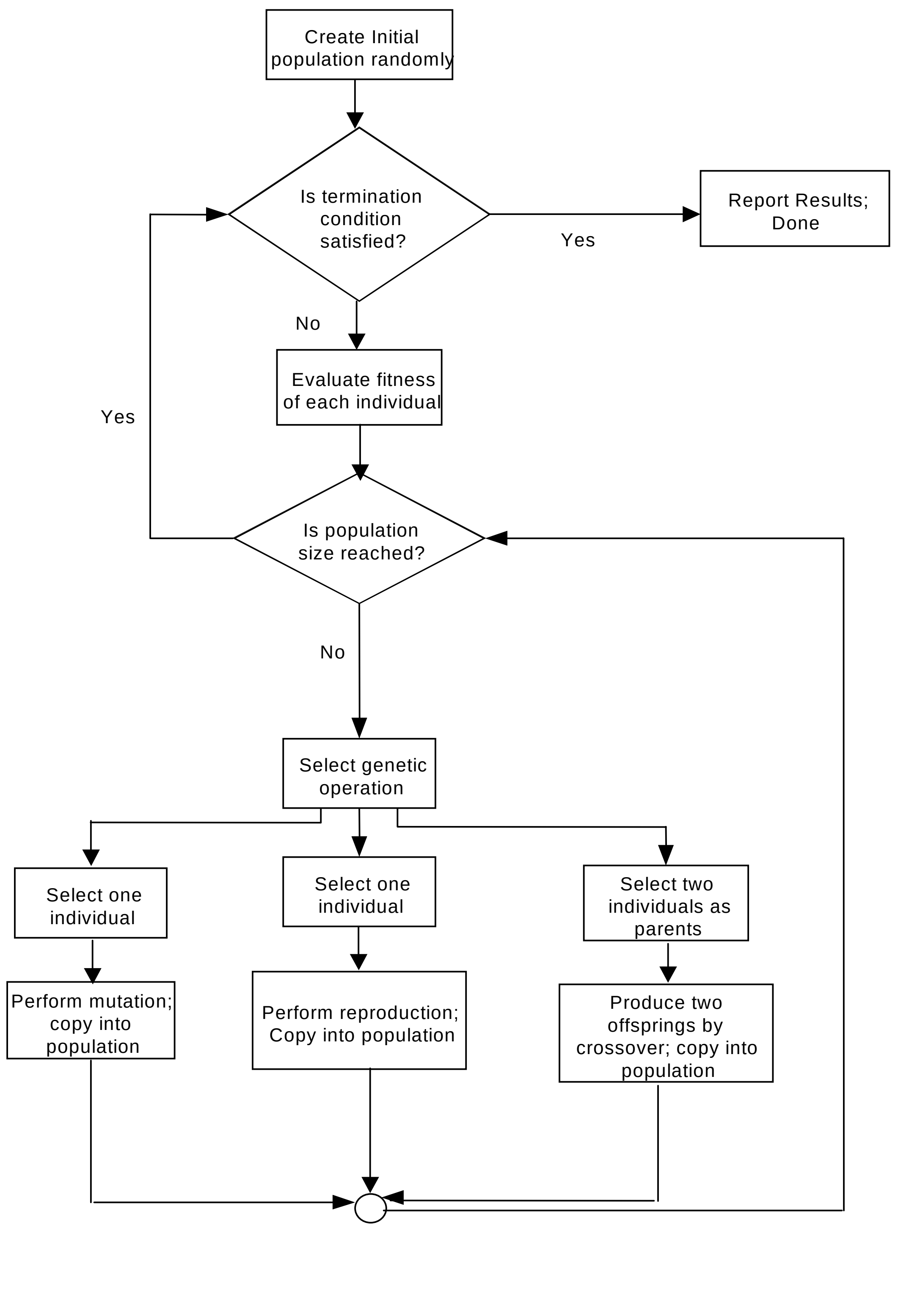
Create Initial population randomly
Is termination condition satisfied?
Yes
Report Results; Done
No
Evaluate fitness of each individual
Yes
Is population size reached?
No
Select genetic operation
Select one individual
Select one individual
Select two individuals as parents
Perform mutation; copy into population
Perform reproduction; Copy into population
Produce two offsprings by crossover; copy into population

Figure 1. The GA Process

Generally, termination occurs when the "optimal" solution is discovered or a certain number of generations are produced.

This process mimics evolution in achieving novelty in its approaches to maintaining fitness [Levy, 1992]. Genetic algorithms generate high quality solutions but have fewer tendencies to terminate on local optima than traditional techniques. Genetic algorithms outperform traditional learning techniques, especially when the solutions that have to be learned are complex. They are especially useful when there is no domain knowledge available to guide the search for solutions [Holsheimer and Siebes, 1994] or when the noisy data is used in data mining [Goldberg, 1994].

GENETIC PROGRAMMING

Building on the principles of GA, Koza introduced genetic programming (GP) [Koza, 1992]. GP uses symbolic expressions (S-expressions)- rather than bit strings -as units being evolved by a genetic program. These S-expressions are essentially subroutines or mathematical functions that are commonly expressed as tree structures. In other words, the S-expressions form tree-shaped "chromosomes." .The objects that make up the population in GP can be thought of as programs that are solutions to the problem at hand. In the context of our paper, the production rules that we are interested in discovering can be represented as these programs.

By starting with a randomly produced generation of such programs and using the principles of evolution discussed above, we can evolve populations of programs that satisfy user-defined criteria of fitness. For example, the crossover operation is the equivalent of swapping branches of two parent trees. While genetic programming is used successfully in robotics, game playing, and discovering mathematical theorems [Hirsh, 2000, Wong et al., 2000], its value as a data mining technique in business is not yet fully explored [Skip, 1999]

IV. DISCOVERING PRODUCTION RULES WITH GPR: METHOD AND IMPLEMENTATION

In this section, we discuss the methodology used by GPR and its implementation.

THE GPR METHOD

GPR is based on the principles of Genetic Programming. It implements the concepts of crossover, reproduction, and mutation in the generation of probabilistic production rules. Each production rule is treated as a program. Adapted from Koza's [1992] GP methodology, GPR generation of rules involves the following steps:

Step 1. Generate an initial population of random composition of functions and terminals[3]of the problem. In the case of GPR, the functions included comparison and range operators.

Step 2. Iteratively perform the following sub-steps until the termination criterion is reached:

- Evaluate each rule in the population and assign it a fitness value based on a fitness function. Various knowledge quality functions discussed in the next subsection are used as fitness functions.
- Create a new population of rules by applying the following operations (based on specified probability distributions for these operations)
    - ♦ Copy existing rules to the new population
    - ♦ Create new rules by genetically recombining randomly chosen parts of existing rules. During the evolution of "solutions", a parent rule or parts of it can be paired with another parent rule or a rule fragment to produce offspring rules

[3] The term terminals is used in GP to refer to the variables and constants used as coefficients and parameters in functions.

♦ Mutate a selected segment of the rules randomly

Step 3. Choose the best rules in the population as the solution.

IMPLEMENTATION

GPR is implemented in the C++ programming language. Utilities for handling input data from flat files, spreadsheets and relational databases are provided. Because of the computation intensive nature of the system, it is implemented in a SUN Sparc computing environment. User-defined parameters are specified in a parameter file that is consulted at the beginning of the run. The program also provides facilities for maintaining checkpoints at periodic intervals (say, at the end of every five generations). If the program is terminated for any reason, it can be restarted from the last saved checkpoint.

A unique strength of GPR as a data-mining tool is its ability to generate rules simultaneously that satisfy any knowledge quality function. GPR can maintain and evolve multiple populations of solutions in parallel. Each population, in turn, can be evolved using different fitness functions. By operating on the same data set and defining different fitness functions for each population, GPR can generate "high quality" rules satisfying the respective fitness functions. This simultaneity provides a user the ability, for example, to generate exact rules, rules with maximum coverage, and rules that find rare occurrences simultaneously.

The search for the best rule is guided by a variety of parameters (commonly used in genetic programs) that are specified by the user. These parameters include the population size, number of generations for evolution to occur, method for growing trees, and method for selection of individuals for probabilistic events such as selection and crossover.

In GPR, the user can also modify the parameters to influence the following:

- The maximum number of attributes that can be included in a rule

- The maximum depth of rules generated
- The number of rules that will be printed from each generation
- The number of populations that must be evolved in parallel
- The number of best rules to be reported at the end of the run

GPR can use a training data set and a test data set for training the system to generate the rules and to ensure the utility of the generated rules in a data set not yet seen by the system, respectively.

GPR OUTPUT

GPR produces production rules of the form

IF <Condition> Then <Target>.

It recognizes the independent variables as the Left-Hand-Side (LHS) and the dependent variable as the Right-Hand-Side (RHS) of a rule. At the end of the run, GPR outputs the evolved rules in the order of their fitness. It should be noted that GPR uses a parse tree representation of programs. Though the program will output only unique tree structures, some of these structures (which contain nested sub-trees), when simplified, may represent the same rule. Therefore, if the user is interested in a set of rules rather than just the best rule, it is advisable to output a larger set of rules so that after simplification, sufficient number of unique rules is available. By default, the program maintains a very large set of rules , ranked according to the fitness function for output. As the population size in genetic programs is typically very large, the size of the population is used as the default size of the set of best rules. In generation 0, this set will contain the same rules as the population. As evolution progresses, the best rules may be found in any of the generations up to the current one.

FITNESS MEASURE IN GPR

The fitness measure is a quantitative evaluation of how well the production rule that was created by GPR matches the data set it is trained and tested upon.

Once the fitness value is determined it is used by the GP package to select which individual programs (production rules) will be used to evolve a new generation of programs.

A variety of measures for evaluating the fitness of discovered knowledge are suggested in the literature. The choice of the measure (called knowledge quality function) determines the characteristics of the knowledge discovered. In contrast to other data mining systems, GPR provides flexibility in that many knowledge quality functions can be easily incorporated to produce production rules that satisfy different needs. For example, the knowledge quality function Rule Interest measures the difference between the actual number of instances where both the description (*y*) and the outcome (*x*) are true, and the number expected if the outcome (*x*) were independent of the description (*y*) [Smyth and Goodman, 1991]. Smyth and Goodman propose the J-measure using information theory concepts [Smyth and Goodman, 1992]. This measure emphasizes the rare outcomes. Certainty is an intuitively appealing measure and is frequently called "strength" or "confidence [Piatetsky-Shapiro et al., 1993] [Matheus et al., 1993].

Table 1 is a contingency table that helps explain a sample fitness function. We can consider an instance in the data set as either conforming (TRUE) or not conforming (FALSE) to the condition set by the LHS of the production rule generated by GPR. We then consider whether that same instance conforms (TRUE) or does not conform (FALSE) to the class set by the RHS. We can perform this same test on every instance in the data set and thereby place each instance in one of four positions in the contingency table.

Table 1. Contingency Table for Fitness Value Computation

	Target Attribute (RHS) is	
	True	**False**
Description (LHS) is True	A {HIT-HIT}	B {MISS-HIT}
Description (LHS) is False	C {HIT-MISS}	D {MISS-MISS}

In essence, quadrant "A" represents the number of instances where the LHS and the RHS are both TRUE; this quadrant is sometimes referred to as the HIT-HIT category. "B", the MISS-HIT category, represents the number of instances where the LHS is TRUE but the RHS is FALSE. It follows that "C" is the HIT-MISS category where the RHS is TRUE but the LHS is FALSE, and that "D" represents the MISS-MISS category where both the LHS and RHS are FALSE. The sum of all instances in categories A, B, C, and D is equal to the total number of instances.

Fitness functions may be defined by using some combination of these four categories to measure how well the generated production rule solves the problem at hand. The simplest fitness functions may take the form:

number of MISS-HITS divided by the number of HIT-HITs

That is, category B divided by category A. Whatever the configuration used for fitness evaluation, GPR attempts to minimize (or maximize, if desired by the user) the fitness value through evolution.

V. TO LEAVE OR NOT TO LEAVE – DISCOVERING EMPLOYEES' HIDDEN DECISION MAKING RULES: A CASE STUDY

Since the end of the cold war, the U.S. Department of Defense (DoD) engaged in a major effort to reduce manpower. While trying not to compromise combat readiness, the Department uses a double strategy for its fighting force.

- Continue aggressively to recruit young men and women, preparing them to be the next generation of war fighters.
- Reduce the number of troops by offering a “voluntary separation incentive” or a “special separation benefit” program (VSI/SSB).

The Marine Corps, one of the four DoD services, applies the Voluntary Separation Incentive / Special Separation Benefit (VSI/SSB) program. It maintains a database of all Marines containing 35 attributes that could potentially explain individual participation behavior. The 35 attributes are arranged in three categories:

- Those that may determine the value of military compensation received by an individual (Four variables: grade, proficiency pay, marital status, and special pay eligibility).
- Those that may determine the potential value of a service member in the civilian job market (12 variables including occupational specialty, education level, intelligence, and experience, and the level of security clearance possessed).
- Non-monetary factors (19 variables including proxies for job satisfaction, promotion opportunity, job security, and physical fitness).

With about 30% of enlistees not completing their first enlistment, the potential for little or no return on initial investment in personnel is quite high. Recruiting and training service personnel is an expensive activity with initial training costs ranging up to $50,000 per person. Therefore, the DoD is interested in gaining insights into the relation between attrition and the economic behavior of its troops. In this section, we use this problem to illustrate the applicability GPR to uncover decision rules from relational databases. Our study specifically addressed the issue of understanding the number of tangible and intangible variables that may influence a service member’s decision to accept a VSI/SSB program.

DATA SET

The Marine Corps database contained 12,787 observations with the 36 attributes (one for target and 35 for description). The descriptor variables contained a mix of both continuous and discrete variables. The target variable (VSI/SSB) specifying whether a marine took advantage of the VSI/SSB program takes discrete values of {Yes, No or Unknown}[4].

INTERPRETATION OF GPR OUTPUT

Figure 2 shows an example of a rule generated by GPR. The rule can be read as follows,

"IF the attribute RACE is equal to 'M', OR the attribute PRESGEO = IL, or the variable FIS_YRGROUP = FY CURRENT and the attribute PFTCL_GP is not equal to C5, THEN the value of VSI/SSB is equal to 'NO".

This rule means that any Marine of Asian/ Pacific descent or any marine stationed in Illinois, or any Marine eligible for the current fiscal year program and not on a medical waiver preventing participation in the physical fitness test did not participate in the VSI/SSB program.

For each rule generated, GPR can also provide a number of statistics including:

- Coverage: The percentage of the database records covered by the LHS.
- Number of misclassified records
- Confidence in the rule generated, based on correct or misclassification of data
- Fitness in Training Data Set
- Fitness in Test Data set

[4] GPR can also handle continuous target attributes.

IF

RACE = M OR

PRESGEO = IL OR

(FTS_YRGROUP = FY CURRENT AND PFTCL_GROUP = NOT C5)

THEN

VSI/SSB = NO

```
Number of records matched by LHS:   3320
Number of misclassified records:      22
Confidence:                       0.9934
```

Figure 2. A Sample Rule Generated by GPR

In the example in Figure 2, of all the instances tested in the database, 3320 matched the above description with respect to the LHS. In addition, 22 of those 3320 matched instances were misclassified and possessed a RHS of something other than 'No'. In other words, 22 of these instances belong to the class 'Yes' or 'Unknown' (*i.e.,* VSI/SSB equal to 'Yes' or 'Unknown'). The confidence value of .9934 (3298/3320) corresponds to the proportion of correctly classified instances according to this production rule.

The focus of the discussion here is to illustrate the ability of GPR to generate rules from real life data. We highlight the ability of GPR to produce rules that satisfy different criteria specified by the user. In Section IV, several knowledge quality functions were proposed to represent the different characteristics of rules that a user may be interested in. We describe the specification of fitness functions corresponding to these knowledge quality functions and the results obtained from GPR.

To evaluate the performance of GPR, we conducted a comparative analysis based on Major and Mantgano's [1993] framework. They propose a methodology to evaluate the interestingness of rules generated by a data mining system. Interestingness is defined according to performance, simplicity, novelty and significance. We used the mushroom database obtained from the University of California at Irvine's Repository of Machine Learning Databases and Domain as a reference database. We compared GPR with popular classification algorithms, i.e., CART, C4.5, Bayes, MML, SMML and Bayes with look-ahead option [Buntine and Niblett, 1992, Mitchell, 1999]. GPR produces 9.5 times as many most general rules. It should be noted that the very nature of the hidden patterns in a database might favor one approach to another. We believe that a variety of techniques need to be used to fully discover and exploit the knowledge hidden in large databases.

A weakness of GPR when compared to other algorithms is the computational complexity of its procedures. Whereas, algorithms such as CART and C4.5 are capable of very rapidly producing results, GPR (as is common with GA based systems) will require several hours of execution to produce its results. This requirement makes this system best suited for off-line applications rather than those where response time is critical. However, with the availability of faster and cheaper storage and processor technologies, this is unlikely to be a major shortcoming for most applications. Further, genetic algorithms and programming lend themselves to parallel processing very easily. Using parallel processing, GRP can be easily modified to exploit the power of a network of computers.

DISCUSSION OF FINDINGS

In our study we used the three knowledge quality functions specified in Section IV, i.e., Rule Interest, J-measure and Certainty. The objective was to study the ability of GPR to produce interesting rules that are based on each of these criteria. The characteristics of the rules are examined to evaluate whether they meet the objectives of these knowledge quality functions.

Evaluation of Discovered Rules

If each of these knowledge quality functions discovers a distinctively different set of rules, it becomes possible to choose the most appropriate of them for different data-mining applications. Table 2 provides some summary statistics for the best 20 rules discovered by the three knowledge quality functions.

Table 2. Summary Statistics on 20 Best Rules

	Knowledge Quality Functions		
	Rule Interest	**J-measure**	**Certainty**
Average coverage [1]	28%	19%	8%
Minimum number of examples covered by a rule	15%	8%	0.1%
Maximum number of examples covered by a rule	35%	34%	12%
Minimum confidence: *p(x|y)*	97%	95 %	100 %
Maximum confidence: *p(x|y)*	100 %	100 %	100 %

[1] *Coverage is measured as a percentage of the records in the database matched by the rule.*

Figure 3 shows sample rules produced using the three measures. To the extent the differences among the sets of rules discovered could be defined, a choice can be made about their appropriateness as knowledge quality functions for data-mining applications.

Certainty

The Certainty function finds a wealth of rules, many of which are exact rules: *e.g.*, rules that predict the outcome without any misclassifications. All exact rules are equally valued by the Certainty function. Certainty discovered a very large number of exact rules, including some that applied to only three or four

<u>Knowledge Quality Function: CERTAINTY</u>

IF

    AGE > 45 AND
    OCCUPATIONL_FIELD = UNDETERMINED

THEN

    VSI/SSB = NO

Number of records matched by LHS: 15

Number of misclassified records: 0

<u>Knowledge Quality Function : J- MEASURE</u>

IF

    32 < Number of months until end of active service < 73

THEN

    VSI/SSB = NO

Number of records matched by LHS: 1193

Number of misclassified records : 65

<u>Knowledge Quality Function : RULE INTEREST</u>

IF

RACE = M OR

PRESGEO = IL OR

(FTS_YRGROUP = FY 94 AND PFTCL_GROUP = NOT C5)

THEN

VSI/SSB = NO

Number of records matched by LHS: 3320

Number of misclassified records: 22

Figure 3: Sample Rules Produced by Three Knowledge Quality Functions

of the 12,787 records in the database. It is possible to provide a minimum coverage for rules to eliminate from consideration rules that applied to only a few examples in the database. For example, a rule discovered by this function (shown in Figure 3) states that “If the age of the Marine is over 45 and has the occupational field undetermined, s/he will not accept VSI/SSB.” This rule, which has a certainty of 100%, is applicable to only 15 records. On closer examination by the user, it turns out that these examples represent a unique set of individuals that were not even considered for the VSI/SSB program. It should be noted, often exact rules that apply to very large proportions of the database are not particularly interesting because they are usually quite obvious. For example, a rule that states, "If the number of years in service in the same rank is high, then the compensation is high" is likely to be obvious to the user. The lowest certainty value of the rules output by the program was 99.8% among the best 100 rules generated by the system. Of these, 20 exact rules that applied to at least 500 examples were generated by this function.

Certainty is a useful knowledge quality function when the rules discovered do not apply to either a preponderance or a very small fraction of the examples in the database. Then, the rules discovered are neither obvious nor trivial.

Rule Interest and J-measure

Both Rule Interest and J-measure are based on information theory concepts and attempt to discover rules with similar characteristics.

Smyth and Goodman observe that Rule Interest will undervalue "rare events" [Smyth and Goodman, 1991]. Figure 3 contains sample rules generated with the two measures. As expected, the sets of "best" rules discovered by Rule Interest and J-measure are similar in many respects. A few exact rules that apply to most of the examples in the database were found by both of these knowledge quality functions, but the other criteria built into these functions led them to find other, more interesting rules. Both functions found many rules that applied to about 10% of the population, while Certainty quality function did not find them. As displayed in Table 2, Rule Interest and J-measure differ primarily in the average proportion of the database covered by the respective sets of rules. All the rules discovered by Rule Interest apply to at least 15% of the database. The descriptions in the rules discovered by J-measure apply to at least 8% of the database. Both functions found rules with descriptions that apply to approximately a third of the examples in the database.

In general, J-measure is able to discover a wider range of rules. It is successful in discovering rules that apply to a small proportion of the examples in the database. This feature would make J-measure a more suitable knowledge quality function for applications where unusual activity is a critical aspect of the knowledge to be discovered; (*e.g.,* modeling unexpected behaviors of a production process or the consumers likely to respond to a unsolicited credit card offer). Rule Interest would be valuable where more general patterns are of interest, such as relationships between income and credit worthiness.

The major difference between the two functions is the complexity of the rules discovered: J-measure tends to favor very simple rules with many containing only one attribute in the description, while the Rule Interest found more complex rules. However, in our case study, the difference between the complexities of rules discovered by the two functions is not large enough to suggest the choice of either function based on this measure. Importantly, both functions generated several interesting rules and several of these rules were discovered by both functions.

Based on our case study and analysis, each knowledge quality function used in our experiment has strengths and weaknesses. To the user, certainty as a knowledge quality function is the easiest to understand. Further, it is likely to find rules that are obvious to the user, thereby increasing the confidence of the user that the system is capable of finding recognizable, general patterns. As stated earlier, these patterns are not very interesting if they do not 'discover' new knowledge. Therefore, other knowledge quality functions have a role in discovering previously unknown knowledge [Padmanabhan and Tuzhilin, 1999].

Rule Interest and J-measure are both based on the same principles of information theory. Therefore they are successful at discovering new and interesting knowledge with often overlapping characteristics. Both functions find rules with high coverage as well as rules with rare outcomes. Though J-measure discovers rules that apply to a smaller proportion of the records in the database than Rule Interest, it also finds simpler rules than Rule Interest.

In general, the choice of a knowledge quality function for a data-mining system is primarily dependent on the needs of the application. If the objective is to discover rules about rare events such as production process failures in a manufacturing setting or the characteristics of consumers that would respond to a direct mail campaign in a consumer database, J-measure would be the preferred knowledge quality function. Rule Interest would be more appropriate if the objective is to find general but not necessarily exact patterns such as finding characteristics of fast moving products in an inventory database. Certainty would be more appropriate if it is critical to have high level of confidence in the knowledge discovered such as in mission critical applications.

The case study illustrates that the approach used in GPR can be used to discover knowledge in real life data mining applications. Further, the ability to simultaneously discover knowledge satisfying various user-defined criteria makes GPR a valuable tool for decision makers.

VI. RELATION OF GPR TO OTHER DATA MINING APPROACHES

Several algorithms developed in the field of machine learning induce decision tress or production rules. ID3 and its descendent C4.5 are among the most popular [Quinlan, 1993] induction based techniques. C4.5 produces a decision tree by recursively splitting the training data into smaller subsets based on some test to be carried out on a single attribute value, with one possible branch and subtree for each possible outcome of that test. The algorithm aims to create a tree that requires a minimum number of tests. It starts with the attribute with the best discriminating power and splits the training data into subsets. This recursive process is continued until all the cases in the training data are classified and/or no other attribute is available for further classification. Various algorithms differ in the kinds of tests they use in partitioning the data. ID3 and C4.5 use the principle of information theory and use information gain and gain ratios for this purpose [Friedman, 1977, Quinlan, 1993]. Other approaches minimize the entropy (a measure of the information content) of the class of distribution on one of the subsets [Cooper and Giuffrida, 2000]. Provost and Kolluri present a survey of methods for scaleable induction algorithms [Provost and Kolluri, 1999].

For the purposes of increasing human understanding some algorithms break down a single large tree into a hierarchy of small trees [Shapiro, 1987]. Algorithms such as C4.5 prefer the generation of production rules similar to the ones produced by GPR. However, C4.5 derives production rules only by converting the decision trees and therefore, with complex trees, the rules can become complex as well. In contrast, the nature of the production rules produced by GPR can be easily controlled using the knowledge quality functions and various parameters on the nature and size of the trees produced. Provost and Buchanan [1995] present a rule learning algorithm that conducts a systematic search of a wide variety of possible rules by considering several independent paths in the search space. However, this approach performs a thorough search only when categorical variables are involved. In contrast, GPR handles continuous data both in the description and target variables.

Genetic algorithms are increasingly popular to solve problems in a variety of domains including engineering, economics, management science and other areas (see [Baack et al., 1997], [De-Jong, 1999] [Holland, 1992] for surveys). Genetic programming was successfully used in automatic induction in a wide variety of applications ranging from generating small subroutines to real-time problems such as robot control. Koza [Koza, 1994, Koza, 2000, Koza et al., 1999] asserts that GP has already achieved the goal of producing results that equal or exceed human performance in a variety of domains such as algorithm design, game playing, pattern recognition, control and design. Taking a high-level statement of a problem's requirements, GP is capable of producing solutions that infringe on or improve on previously issued patents in problems like circuit design [Koza et al., 1999]. The application of GP to solving business problems is exemplified by a system developed by Dworman, Kimbrough, and Laing [Dworman et al., 1995] that discovers high quality negotiation patterns in a multi-agent game. This system uses simulations to evolve negotiation patterns in the form of if-then-else rules. In GPR, our interest is in inducing (more) comprehensive rules from training data. The ability of discovering knowledge from databases using GP is illustrated in the LOGENPRO system [Wong et al., 2000]. This system learns logic programs from noisy databases. However, the applications of GA/GP in knowledge discovery and data mining, especially for discovering production rules, have been very limited [Dhar et al., 2000]. Holland [1992] establishes the appropriateness of using Genetic Algorithms for learning such patterns. The GLOWER system [Dhar et al., 2000] represents an approach to learning rules from data using genetic algorithms. The system exploits the power of GA to scour the search space thoroughly in finding interesting rules. This approach is especially useful when the search space includes continuous variables. Unlike greedy search processes used by machine learning systems, Genetic Algorithms are less constrained in searching for all applicable rules. In GLOWER, the chromosomes are made up of sets of genes that represent constraints on a single descriptor variable. Although GPR also uses the power of genetic search, it differs from GLOWER in a number of ways. First, GPR uses genetic programming to represent production rules rather than bit-strings used in GA-based systems such as GLOWER. Because production rules are simple programs, genetic programming provides a natural representation for these. Further, whereas GLOWER uses conjunctions (AND) of the sets of genes representing variables, GPR may use conjunctions, disjunctions, and negations (AND, OR, NOT), thereby enlarging the search. Similar to GPR, GLOWER also supports the user of different fitness functions with a single population. As both the systems are essentially based on genetic search, GPR can benefit from the use a variety of search heuristics (such as entropy reduction and sequential niching) that have been demonstrated to be successful in producing production rules with better support (compared to tree induction algorithms).

VII. CONCLUSIONS

This article presents a novel approach to data mining using the principles of genetic programming to generate production rules from a database. This system is based on the principles of genetic programming, a variation of genetic algorithms. Our GPR prototype system possesses the unique ability to discover knowledge that satisfies a variety of user-defined criteria specified in knowledge quality functions. Unlike most datamining algorithms that find patterns that fit a single criterion, GPR can be used to find patterns that satisfy multiple (and possibly conflicting) objectives simultaneously. For example, one population can be used to discover rules with high certainty, whereas another can simultaneously be used to discover more general rules and yet another can discover rules that discover rare occurrences. The ability to discover rules that satisfy user defined criteria of 'interestingness' is especially important in domains where the user may not have a clear preference for the characteristics of the hidden knowledge.

To illustrate the method, the system was used in a data mining application for the Marines Corps. It produced interesting results. Our system is also shown to perform well when compared to popular classification algorithms. The approach presented here is sufficiently generic to offer wide applicability to a large number of potential data mining applications. With the information revolution enabled by the World Wide Web, organizations are collecting enormous amounts of data from diverse sources ranging from web clickstreams, e-commerce transactions [Adomavicius and Tuzhilin, 2001], scientific (e.g., space and geological) exploration [Valdes-Perz, 1999] to mapping the human genome [Steffen, 1999]. Systems such as GPR are essential to understanding the hidden knowledge in these vast sources of data. As our approach does not require any domain specific knowledge, it can be used without major modification in different domains. The potential for applications of this technology in business domains such as one-to-one marketing, fraud detection, customer-relationship-management, cross and up selling are numerous and growing.

Editor's Note: This article was received on September 18, 2000. It was with the authors four weeks for one revision and was published on April 5, 2000

ACKNOWLEDGEMENTS

This research was supported in part by a grant from the U.S. Air Force Research Laboratory and the Department of Defense DAU EARP program.

REFERENCES

Adomavicius, G. and A. Tuzhilin (2001) "Using Data Mining Methods to Build Customer Profiles," *IEEE Computer* (34)2, pp. 74-82.

Agrawal, R., T. Imielinski, and A. Swami (1993) "Database Mining: A Performance Perspective," *IEEE Transactions on Knowledge and Data Engineering* (5)6, pp. 914 -925.

Baack, T., D. Fogel, and Z. Michaelewicz (1997) *Handbook of Evolutionary Computation*. New York, NY: Oxford University Press.

Buntine, W. L. and T. Niblett (1992) "A Further Comparison of Splitting Rules for Decision-Tree Induction," *Machine Learning* (8)1, pp. 75-85.

Chung, H. M. and P. Gray (1999) "Special Section: Data Mining," *Journal of Management Information Systems* (16)1, pp. 11-16.

Cooper, L. G. and G. Giuffrida (2000) "Turning Datamining into a Management Science Tool: New Algorithms and Empirical Results," *Management Science* (46)2, pp. 249-264.

De-Jong, K. A. (1999) "Evolutionary Computation for Discovery," *Communications of the ACM* (42)11, pp. 51-53.

Dhar, V. (1998) "Data Mining in Finance: Using Counterfactuals to Generate Knowledge from Organizational Information Systems," *Information Systems* (23)7, pp. 423-437.

Dhar, V., D. Chou, and F. Provost (2000) "Discovering Interesting Patterns for Investment Decision Making with Glower -- a Genetic Learner Overlaid with Entropy Reduction," *Data Mining and Knowledge Discovery* (4)4, pp. 251-280.

Dhar, V. and A. Tuzhilin (1993) "Abstract-Driven Pattern Discovery in Databases," *IEEE Transactions on Knowledge and Data Engineering* (5)6, pp. 926 -938.

Dworman, G., S. O. Kimbrough, and J. D. Laing (1995) "On Automated Discovery of Models Using Genetic Programming: Bargaining in a Three-Agent Coalition Game," *Journal of Management Information Systems* (12)3, pp. 97-125.

Frawley, W. J., G. Piatetsky-Shapiro, and C. J. Matheus (1991) “Knowledge Discovery in Databases: An Overview,” in G. Piatetsky-Shapiro and W. Frawley (Eds.) *Knowledge Discovery in Databases*, Menlo Park, CA: AAAI Press.

Friedman, J. H. (1977) "A Recursive Partitioning Decision Rule for Nonparametric Classification," *IEEE Transactions on Computers* (26)4, pp. 404-408.

Goldberg, D. E. (1994) "Genetic and Evolutionary Algorithms Come of Age," *Communications of the ACM* (37)3, pp. 113-119.

Grefenstette, J. J. (1988) "Credit Assignment in Rule Discovery Systems Based on Genetic Algorithms," *Machine Learning* (3)2-3, pp. 225-245.

Hirsh, H. (2000) "Genetic Programming," *IEEE Intelligent Systems & Their Applications* (15)3, pp. 74-84.

Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press.

Holland, J. H. (1992) *Adaptation in Natural and Artificial Systems. An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA: MIT Press.

Holsheimer, M. and A. Siebes. (1994) *Data Mining: The Search for Knowledge in Databases*. CWI Amsterdam, The Netherlands. Technical Report CS-R9406.

Jorng-Tzong, H. and Y. Ching-Chang (2000) "Applying Genetic Algorithms to Query Optimization in Document Retrieval," *Information Processing & Management* (36)5, pp. 737-759.

Koza, J. (1994) *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, MA: MIT Press.

Koza, J. (2000) "Human-Competitive Machine Intelligence by Means of Genetic Programming," *IEEE Intelligent Systems* (15)3, pp. 76-878.

Koza, J. R. (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press.

Koza, J. R., F. H. Bennett, M. Keane, and D. Andre (1999) *Genetic Programming III: Darwinian Invention and Problem Solving*. San Mateo, CA: Morgan Kaufmann.

Levy, S. (1992) *Artificial Life*. New York: Vintage Books.

Major, J. A. and J. J. Mantgano. (1993) “Selecting among Rules Induced from a Hurricane Database.” *AAAI Workshop on Knowledge Discovery in Databases, Menlo Park, CA*.

Matheus, C. J., P. K. Chan, and G. Piatetsky-Shapiro (1993) "Systems for Knowledge Discovery in Databases," *IEEE Transactions on Knowledge and Data Engineering* (5)6, pp. 903 -913.

Michalewicz, Z. (1996) *Genetic Algorithms + Data Structures = Evolution Programs*. New York, NY: Springer-Verlag.

Mitchell, T. M. (1999) "Machine Learning and Data Mining," *Communications of the ACM* (42)11, pp. 30-36.

Nilsson, N. J. (1998) *Artificial Intelligence: A New Synthesis*. San Mateo, CA: Morgan Kaufmann.

Padmanabhan, B. and A. Tuzhilin (1999) "Unexpectedness as a Measure of Interestingness in Knowledge Discovery," *Decision Support Systems* (27)3, pp. 303-318.

Piatetsky-Shapiro, G. (1999) "The Data-Mining Industry Coming of Age," *IEEE Intelligent Systems & Their Applications* (14)6, pp. 32-34.

Piatetsky-Shapiro, G., C. J. Matheus, P. Smyth, and R. Uthurusamy (1993) “Progress and Challenges in Knowledge Discovery in Databases,” in G. Piatetsky-Shapiro and W. Frawley (Eds.) *Knowledge Discovery in Databases*, Menlo Park, CA: AAAI Press.

Provost, F. J. and B. G. Buchanan (1995) "Inductive Policy: The Pragmatics of Bias Selection," *Machine Learning* (20)1-2, pp. 35-61.

Provost, F. J. and V. Kolluri (1999) "A Survey of Methods for Scaling up Inductive Algorithms," *Data mining and knowledge discovery* (3)2, pp. 131-169.

Quinlan, J. R. (1986) “The Effect of Noise on Concept Learning,” in R. S. Michalski, J. G. Carbonell, and T. Mitchell (Eds.) *Machine Learning II: An Artificial Intelligence Approach*, Los Altos, CA: Morgan Kauffmann.

Quinlan, J. R. (1988) “Simplifying Decision Trees,” in B. Gaines and J. Boose (Eds.) *Knowledge Acquisition for Knowledge-Based Systems*, London: Academic Press.

Quinlan, J. R. (1993) *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kauffman.

Shapiro, A. D. (1987) *Structured Induction in Expert Systems*. Wokingham, UK: Addison-Wesley.

Skip, D. (1999) "Refocusing Artificial Intelligence Research on Real-World Applications," *Research & Development* (41)5, pp. 46-48.

Smyth, P. and R. M. Goodman (1991) “Rule Induction Using Information Theory,” in G. Piatetsky-Shapiro and W. Frawley (Eds.) *Knowledge Discovery in Databases*, Menlo Park, CA: AAAI Press.

Smyth, P. and R. M. Goodman (1992) "An Information Theoretic Approach to Rule Induction from Databases," *IEEE Transactions on Knowledge and Data Engineering* (4)4, pp. 301-316.

Steffen, S.-K. (1999) "Discovery in the Human Genome Project," *Communications of the ACM* (42)11, pp. 62-64.

Valdes-Perz (1999) "Discovery Tools for Science Applications," *Communications of the ACM* (42)11, pp. 37-41.

Wang, J. (2000) "Trading and Hedging in S&P 500 Spot and Futures Markets Using Genetic Programming," *The Journal of Futures Markets* (20)10, pp. 911-942.

Wong, M. L., K. S. Leung, and J. C. Y. Cheng (2000) "Discovering Knowledge from Noisy Databases Using Genetic Programming," *Journal of the American Society for Information Science* (51)9, pp. 870-881.

Zytkow, J. M. (1993) "Introduction: Cognitive Autonomy in Machine Discovery," *Machine Learning* (12)1-3, pp. 7-16.

ABOUT THE AUTHORS

Balasubramaniam Ramesh is Associate Professor of Computer Information Systems at Georgia State University. Ramesh received his Ph.D. degree in Information Systems from New York University. His research work appears in leading conferences and journals including the *IEEE Transactions on Software Engineering, Communications of the ACM, IEEE Computer, IEEE*

Internet Computing, IEEE Intelligent Systems, Annals of Software Engineering, and Annals of Operations Research. His research interests include data mining and supporting knowledge work in the areas such as requirements engineering and traceability in systems development, concurrent engineering, e-services and new product development. His work has been supported by grants from several leading funding organizations including the National Science Foundation, Office of Naval Research, Air Force Research Laboratory, DARPA, and Microelectronic Computer Technology Corporation.

Tung Bui is Matson Navigation Company Professor of Global Business at the University of Hawaii, Manoa. Bui is also director of the Pacific Research Institute for Information Systems Management (PRIISM) and co-director of the Asia-Pacific Economic Cooperation (APEC) Study Center. He is the author of 7 books and over 140 papers. His current research interests focus on effective management of large organizations, electronic commerce, sustainable development, and in collaborative technology, including group decision and negotiation support systems. Professor Bui is a regular consultant and advisor to both governmental and private organizations on a number of public policies that include national and regional planning. Recent activities are described at his website: http://ec.cba.hawaii.edu/tbui.