Genetic programming for knowledge discovery in chest pain diagnosis

Celia C. Bojarczuk[1], Heitor S. Lopes[2], Alex A. Freitas[3]

[1, 2] CEFET-PR / CPGEI, Av. 7 de setembro, 3165. Curitiba – PR, 80230-901. Brazil.
{bojarczuk, hslopes}@cpgei.cefetpr.br
[3] PUC-PR, PPGIA-CCET, Av. Imaculada Conceicao, 1155. Curitiba – PR, 80215-901. Brazil.
alex@ppgia.pucpr.br, http://www.ppgia.pucpr.br/~alex

Abstract

This work aims at discovering classification rules for diagnosing certain pathologies. These rules are capable of discriminating among 12 different pathologies, whose main symptom is chest pain. In order to discover these rules we have used genetic programming as well as some concepts of data mining, with emphasis on the discovery of comprehensible knowledge. The fitness function used combines a measure of rule comprehensibility with two usual indicators in medical domain: sensitivity and specificity. Results regarding the predictive accuracy of the discovered rule set as a whole and the predictive accuracy of individual rules are presented and compared to other approaches.

INTRODUCTION

To classify and diagnose some pathology, one must verify which predicting attributes are most associated with that disease. In this work there are 165 predicting attributes and 12 different diseases (classes) whose main characteristic is the chest pain. The predicting attributes refer not only to the characteristics of the chest pain reported by the patient, but also to other symptoms, signals observed by the physician, details of clinical history and results of laboratory tests. The diseases are: stable angina, unstable angina, acute myocardial infarction, aortic dissection, cardiac tamponade, pulmonary embolism, pneumothorax, acute pericarditis, peptic ulcer, esophageal pain, musculoskeletal disorders, psychogenic chest pain. The goal is to predict which of those diseases a patient has, given the values of the 165 predicting attributes for the patient.

Chest pain is a symptom related to several diseases of cardiovascular, pulmonary, esophageal, psychogenic and other origins. According to the World Health Organization, only cardiovascular diseases represent about 25 % of death rates in the whole world, especially in developed countries. A fast and effective evaluation of chest pain, especially in the emergency room, is a critic problem daily faced by clinicians. The main problem is a prompt discrimination among diseases that could involve life threatening from other less serious pathologies. Clinicians have to diagnose running against time, sometimes under pressure and using little available information. Under this situation, an accurate diagnosis may be fairly difficult. Furthermore, the clinician has to consider factors like: large variability of symptoms and signals, large number of possible diagnostics, little correlation between pain location and its origin, etc. To cope with this diagnostic problem, a number of intelligent systems based on different paradigms were proposed - see, for instance [1,14,16]. Most of these approaches were concerned with diagnosis, but not with comprehensibility of the knowledge that has led to it. Notwithstanding, the availability of a clinical database of chest pain patients has permitted the development of the current work, whose aim is to discover high-level, comprehensible knowledge about the diagnosis of chest pain. Therefore, our work follows the spirit of the relatively recent area of data mining and knowledge discovery, where the goal is to discover knowledge that not only has a high predictive accuracy but also is comprehensible to users [5,7]. Therefore, the user can understand the system's results and combine them with his/her knowledge to make a well-informed decision, rather than blindly trusting the incomprehensible output of a "black box" system.

This work addresses the above classification problem in the context of data mining. In this context the discovered knowledge is often expressed as IF-THEN prediction (classification) rules. The IF part of the rule contains a logical combination of conditions on the values of predicting attributes, whereas the THEN part contains the predicted pathology (class) for a patient whose clinical attributes satisfy the IF part of the rule.

The use of genetic programming (GP) for discovering comprehensible classification rules, in the spirit of data mining, is a relatively underexplored area. We believe this is a promising approach, due to the effectiveness of GP in searching very large spaces [12,13] and its ability to perform an open-ended, autonomous search for logical combinations of predicting attribute values.

This paper is organized as follows: the next two sections aim to give some background to the reader about fundamentals of data mining techniques and the genetic programming paradigm. Next, the proposed system for knowledge discovery using genetic programming is presented in detail. Further, computational results are shown using the system with a medical data base of patients with chest pain. Finally, in the last section results are discussed and conclusions presented.

AN OVERVIEW OF DATA MINING

In essence, the goal of data mining is to discover knowledge from real-world data sets. While there is no universally accepted definition of knowledge, we can mention some properties that the discovered knowledge should have, as follows.

First, the discovered knowledge should be accurate. This refers to the ability of the discovered knowledge in accurately predicting the values of some attribute(s) - or feature(s) - for data that *was not seen* during the run of the data mining algorithm. This is also called the generalization performance of the discovered knowledge. Several ways of measuring generalization performance are discussed in [9].

Second, the discovered knowledge should be comprehensible to the user. The motivation for this is to give the user a solid basis for making better decisions. We are assuming here that data mining is being used as a decision-support system, but the actual decision will eventually be made by a human user. This assumption seems to be appropriate for medical domains where lives are at stake. Note that knowledge comprehensibility is a very subjective concept, unlike predictive accuracy, which can be more easily measured in an objective way. Frequently, the comprehensibility of the discovered knowledge is associated with its syntactical simplicity. Some relevant discussions about knowledge comprehensibility and simplicity can be found in [4, 18].

Third, the discovered knowledge should be somehow interesting and useful. Interestingness and usefulness are probably still more difficult to be measured than comprehensibility. This is currently an active research area, and some measures of knowledge interestingness have been recently proposed [6, 17]. In this paper we will evaluate the quality of the discovered knowledge with respect to both predictive accuracy and comprehensibility, but not interestingness.

There are several data mining tasks, such as classification, clustering, dependence modeling, association, summarization, etc. A review of all these tasks is beyond the scope of this paper. The reader interested in an overview of several data mining tasks is referred to [5,7]. Here we just describe the classification task [9], which is the most studied in the literature and is also the focus of this paper.

In this task each data instance (sometimes called a case, or yet an example) belongs to a class, among a predefined set of classes. The class of an instance is given by the value of a user-specified goal attribute. Instances consist of a set of predicting attributes (features) and a goal attribute. This latter is a categorical (or nominal, or yet discrete) attribute, i.e., it can take on a value out of a small set of discrete values.

In the classification task the aim is to discover some kind of relationship between the predicting attributes and the goal, so that the discovered knowledge can be used to predict the class (goal-attribute value) of a new, unknown-class instance.

There are several paradigms of data mining algorithms, including rule induction, instance-based learning (or nearest neighbors), neural networks, evolutionary algorithms and many other paradigms. In this paper the focus is the paradigm of evolutionary algorithms, particularly genetic programming. It should be noted we cannot claim that one of the above paradigms is superior to the others with respect to predictive accuracy. Each of these paradigms contains many different algorithms, and the predictive accuracy associated with an algorithm depends very much on the data being mined. This fact has been shown both empirically and theoretically [4].

EVOLUTIONARY ALGORITHMS

Genetic Algorithms (GAs), and Genetic Programming (GP), belong to a class of optimization techniques broadly called evolutionary algorithms. GAs were invented by John Holland in the late 1960s, and in his pioneering monograph [10] he established the main theoretical grounding for GAs. Evolutionary algorithms are inspired by the evolution of living organisms and by the Darwinian principle of natural selection, where the fittest individuals have a better chance to survive. Good textbooks about theoretical and practical applications of GAs can be found in [8,11] and about GP in [2,12,13].

In GAs, there is a mapping between the genotype (internal representation of the parameters of the problem) and the phenotype (what the parameters really mean for the problem). Usually, the parameters of the problem are encoded in the genotype as a fixed-size string of bits (or, sometimes, a string of integers). This representation is suitable for a large range of problems, but is very limiting when the nature of the problem requires variable-length or even more sophisticated structures as the shape of the solution.

The representation of an individual is the main difference between GAs and GP. In GP, individuals are represented with variable-length hierarchical structures called “programs”, usually in the form of a tree. In GP, the shape, size and structural complexity of the solution are not limited a priori. This characteristic is an advantage and a drawback at the same time. It is an advantage over GAs because complex structures can be represented. These structures are capable of performing not only mathematical and logical operations, but also iteration, recursion and conditional branching. On the other hand, the representation induces an infinite search space, which requires, besides computational power, more intelligent and adaptive techniques. Despite this problem, GP has been successfully used for a large range of problems, mainly in the engineering and computer science areas.

A brief overview of genetic programming

In GP, the basic idea is the evolution of a population of "programs", i.e., candidate solutions to the specific problem in hand. A program (an individual of the population) is usually represented as a tree, where the internal nodes are functions (operators) and the leaf nodes are terminal symbols. More complex representations, like graphs, are unusual since they require specialized genetic operators. Both the function set and the terminal set must contain symbols appropriate for the target problem. The function set can contain arithmetic operators, logic operators, mathematical functions, etc; whereas the terminal set can contain the variables (attributes) of the problem.

Each individual of the population is evaluated with respect to its ability to solve the target problem. This evaluation is performed by a fitness function, which is problem-dependent. In general, fitness is computed by means of a well-defined mathematical function (or even a complex procedure) that assigns a scalar value to it. In most cases, fitness is evaluated over a set of different representative situations (called fitness cases), sampled from the problem space.

Individuals undergo the action of genetic operators such as reproduction and crossover. These operators are very briefly described as follows - a more detailed analysis about these and other genetic operators can be found in [12].

The reproduction operator selects one individual of the current population in proportion to its fitness value, so that the fitter an individual is the higher the probability that it will take part in the next generation of individuals. After selection, the individual is copied into the new generation. Reproduction reflects the principle of natural selection and survival of the fittest.

The crossover operator replaces a randomly selected subtree of an individual with a randomly chosen subtree from another individual. The application of crossover is independent of the tree topology since it has to meet the closure property of GP. This property states that any function has to accept as operand any possible combination of terminals and functions. Figure 1 illustrates the application of the crossover operator to a couple of parents. The crossover point was chosen at random for both parents. Translating the trees into logical expressions, one can realize the exact results, as follows:

- Parent 1: *(not A) or (B and C)*
- Parent 2: *(D or (not E)) or ((not F) and (not G))*
- Offspring 1: *((not F) and (not G)) or (B and C)*
- Offspring 2: *(D or (not E)) or (not A)*

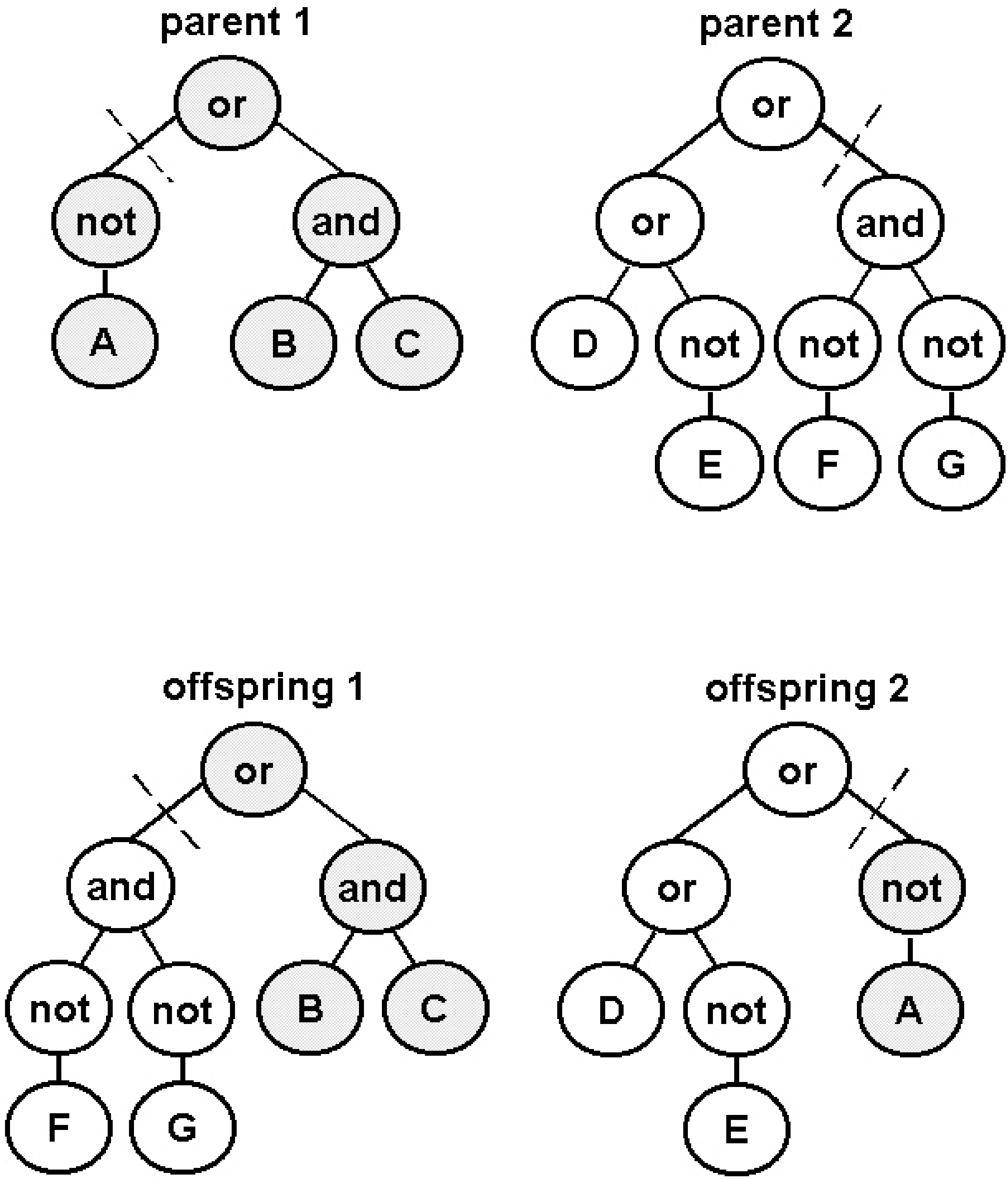

Figure 1: Crossover operator in action. Above: original parents and crossover points. Below: offspring after crossover.

Once genetic operators have been applied to the population according to given probabilities, a new generation of individuals is created. These newly created individuals are evaluated by the fitness function. The whole process is iteratively repeated, for a fixed number of generations or until other termination criterion is met. The result of genetic programming (the best solution found) is usually the fittest individual produced along all the generations.

A GP run is controlled by several parameters, some numerical and some qualitative [12]. However, this is implementation-dependent, i.e., not all GP systems have to follow the same directives, specially those parameters concerning the genetic operators. If special genetic operators are defined, specific parameters have to be set to control their use throughout the run.

Summarizing, the use of GP to solve real-world problems encompasses the previous definition of the following:

- the structural components of the solution, i.e., the set of functions and terminals;
- the generation of the initial population of solutions;
- the genetic operators used to modify stochastically a given solution;
- the measure of goodness of a given solution (fitness function);
- the criteria for terminating the evolutionary process and for designating the result;
- the major and minor parameters that control the GP.

THE PROPOSED GENETIC PROGRAMMING SYSTEM

Function and terminal sets

The terminal set consists of the previously cited 165 binary attributes used in the diagnosis of chest pain. The function set consists of three logic functions, namely AND, OR and NOT. These functions were chosen due to our desire for discovering high-level, comprehensible rules, as mentioned before. Therefore, an individual (program) consists of a combination of the above three logic operators applied to some predicting attributes. Each individual encodes the IF part of a rule, but not the THEN part (the predicted class). The reason for this is the fact that in a given run of the GP all individuals represent rules predicting the same class, as follows.

Each run of our GP solves a two-class classification problem, where the goal is to predict whether or not the patient has a given disease. Therefore, to generate classification rules for the 12 classes, we need to run the GP 12 times. In the first run the GP would search for rules predicting class 1; in the second, for class 2, and so on. When the GP is searching for rules predicting a given class, all other classes are effectively merged into a large class, which can be conceptually thought of as meaning that the patient does not have the disease predicted by the rule. Since the THEN part of the rule does not need to be encoded into the individual, from now on we will refer to each individual as a rule, for the sake of brevity and simplicity.

GP control parameters

In all the experiments reported in this paper the initial population was randomly generated by the well-known ramped half-and-half method, which creates an equal number of trees for tree-depth values varying from 2 to the maximum depth D_i =10. During the run, the maximum tree depth (D_c) was set to 17. The criterion for terminating the run was the maximum number of generations, $G = 50$ (not including the first, randomly generated population). The best individual ever found (see the next subsection) in the run was designated as the result. The population size (M) was set to 500 individuals, the probability of crossover (p_c) and reproduction (p_r) were respectively set to 0.9 and 0.1 and the selection method used for both parents was fitness proportionate. All the parameters used were set to the standard values proposed by Koza [12], who has found that they are suitable for most applications.

Fitness function

The fitness function evaluates the quality of each rule (individual). In this work, the fitness function was based on that proposed by [15]. Before the fitness function is defined, it is necessary to recall a few basic concepts on classification-rule evaluation. When using a rule for classifying a given patient case, depending on the class predicted by a rule and on the true class of the patient, four types of results can be observed for the prediction, as follows:

- true positive (*tp*) - the rule predicts that the patient has a given disease and the patient does have that disease;
- false positive (*fp*) - the rule predicts that the patient has a given disease but the patient does not have it;
- true negative (*tn*) - the rule predicts that the patient does not have a given disease, and indeed the patient does not have it;
- false negative (*fn*) - the rule predicts that the patient does not have a given disease but the patient does have it.

The fitness function used in this work combines two indicators that are commonplace in the medical domain, namely the sensitivity (*Se*) and the specificity (*Sp*), defined as follows:

$$Se = tp \,/\, (tp + fn) \tag{1}$$

$$Sp = tn \,/\, (tn + fp) \tag{2}$$

In fact, GP does not produce necessarily simple solutions. Considering that the comprehensibility of a rule is inversely proportional to its size, something has to be done to enforce GP to produce rules as short as possible. Therefore, we define a measure of simplicity (*Sy*) of a rule, given in equation 3:

$$Sy = (maxnodes - 0.5\ numnodes - 0.5) \,/\, (maxnodes - 1) \tag{3}$$

where *numnodes* is the current number of nodes (functions and terminals) of an individual (tree), and *maxnodes* is the maximum allowed size of a tree (that was set to 65). Equation 3 produces its maximum value of 1.0 when a rule is so simple that it contains just one term. The equation value decreases until its minimum value of 0.5, which is produced when the number of nodes equals the maximum allowed. The reason to set the lower bound to 0.5 is to penalize large-sized individuals without forcing them to disappear. This is specially important in the former generations of a run when most individuals will have very low predictive accuracy, but can carry good genetic material capable of further improvement by means of the genetic operators.

Finally, the fitness function used by our GP is defined as the product of the indicators of predictive accuracy and simplicity:

$$fitness = Se\ .\ Sp\ .\ Sy \tag{4}$$

Therefore, the goal of our GP is to maximize both the *Se* and the *Sp*, and minimize the rule size simultaneously. This is an important point, since it would be relatively trivial to maximize the value of one of these indicators at the expense of significantly reducing the values of the others. Furthermore, the above fitness function has the advantages of being simple and returning a meaningful, normalized value in the range [0..1]. For further analysis of the motivation for maximizing the product *Se.Sp*, regardless of rule simplicity and independently of any evolutionary algorithm, see [9].

COMPUTATIONAL RESULTS

The experiments reported here were done using a modified version of the Lil-GP system, version 1.02 [20] and the data set was the same as used in [14]. The data set consists of 138 examples (patients) and 165 attributes. As usual in the classification literature, the data set was randomly partitioned into two sets, a training set (90 examples) and a test set (48 examples). The experiments were done with a single training/test set partition, rather than doing cross validation, to make our results comparable to the ones reported in [14]. Therefore, we have used the same training/test set partition as the one used in the above reference.

Note that, in principle, this can be considered a difficult classification problem. The reason is that the problem has a very high dimensionality. The problem can be mapped to a 165-dimensional problem, corresponding to the 165 attributes, but unfortunately there are only 90 training examples. This small "density" of the data – i.e., the small value of the ratio of the number of examples over the number of attributes – increases the probability that the discovered set of rules be unduly overfitted to the data. Ideally, the number of examples should be much larger then the number of attributes [9].

For each of the 12 classes, the GP was run ten times, as mentioned before, and the best individual (rule) was the one with the best performance on the training set. For each class we selected the best rule out of the ten rules found in the ten GP experiments.

Therefore, the final result of our experiments is a set of 12 rules, one for each class. We analyzed the quality of these discovered rules in three ways, namely by evaluating the predictive accuracy of the rule set as a whole, the predictive accuracy of individual rules and the comprehensibility of the rules. These analyses are discussed in the next three subsections, respectively.

We have further compared the predictive accuracy of the rule set discovered by our GP system with the predictive accuracy of the rule set discovered by C5.0, a state-of-the-art rule induction algorithm [19]. The result of this comparison is reported in the next subsection.

Predictive accuracy of the rule set as a whole

Among the several different criteria that can be used to evaluate the predictive accuracy of a discovered rule set, we have used the well-known accuracy rate, which, despite its defects [9], seems to be still the most used in the classification literature. The accuracy rate is simply the ratio of the number of correctly-classified test examples over the total number of test examples.

Our GP achieved an accuracy rate of 87.5 % - i.e., the discovered rule set correctly classified 42 out of 48 examples.

We have also applied a rule induction algorithm, C5.0, to the same training / test set partition used in the experiments with our GP system. The algorithm C5.0 (with its default parameters) generated a decision tree that is equivalent to 17 rules, which achieved an accuracy rate of 79.2 % - i.e., it correctly classified 38 out of 48 examples.

Predictive accuracy of individual rules

Although reporting the predictive accuracy of the rule set as a whole is commonplace in the literature, in practice the user will analyze or consider one rule at a time. Therefore, it makes sense to report the accuracy rate of each individual rule. Recall that this is feasible, in our case, because our system discovered only 12 rules (one for each class). Such an analysis of the predictive accuracy of individual rules is less feasible in cases where the data mining algorithm discovers too many rules.

Table 1: Predictive accuracy and simplicity of individual rules.

class #	*Sensitivity (Se)*	*Specificity (Sp)*	*Se . Sp*	*Simplicity (Sy)*
1	0.75	0.97	0,73	0,96
2	1.00	0.93	0.93	0.98
3	1.00	0.97	0.97	0.98
4	1.00	0.97	0.97	1.00
5	0.75	0.97	0.73	0.98
6	0.80	1.00	0.80	1.00
7	1.00	1.00	1.00	1.00
8	0.50	0.95	0.47	0.96
9	1.00	0.95	0.95	0.98
10	1.00	0.90	0.90	0.97
11	1.00	1.00	1.00	0.96
12	0.60	1.00	0.60	0.96
average	**0.86**	**0.96**	**0.83**	**0.97**

Table 1 reports the figures concerning the predictive accuracy of each of the 12 discovered rules. Each row of this table refers to the best rule found in the ten GP runs performed for the corresponding class number. All the figures reported in this table refer to the performance of the discovered rules on the test set. Class numbers refer to those diseases listed in the Introduction section, in the order they appear.

Overall, the discovered rules have both high sensibility (*Se*) and high specificity (*Sp*), which leads to a high value of the product *Se.Sp*. In particular, seven rules (those predicting classes 2, 3, 4, 7, 9, 10, 11) have both *Se* and *Sp* (as well as their product) greater than or equal to 0.90. However, some rules are not so good. For instance, the rule predicting class 8, despite its high value of *Sp*, is less likely to be considered as truly reliable by the user, due to its very low *Se*. This is consistent with the results reported by [14], since class 8 is indeed the most difficult one to predict.

As can be seen in the last row of Table 1, the average value of the product *Se.Sp* is 0.83. This value is slightly smaller than the average value of 0.85 reported in [14] for the same data set, but using another kind of classification algorithm. However, our GP system found comprehensible rules, whereas the algorithm used in [14] works as a black box.

Comprehensibility of the discovered rules

As mentioned in the Introduction, in this work we are interested not only in the predictive accuracy of the discovered rules, but also in the comprehensibility of the rules - in the spirit of data mining.

Rule comprehensibility is significantly more difficult to measure in an objective way, in comparison with predictive accuracy. There is a considerable subjective aspect in the former. In most of the literature, however, rule comprehensibility is associated with syntactical complexity. Usually, the smaller the number of rules and the shorter (the smaller the number of conditions of) the rules the better. In this paper we also follow this principle for evaluating the comprehensibility of the discovered rules.

Concerning the number of discovered rules, our GP system (by design) provides the best possible result, that is exactly one rule for each class. This minimization of the number of discovered rules saves the potentially precious time of the human user, who is required to analyze only the very best rule found for each class. In contrast, some rule induction algorithms may overload the user with a large number of discovered rules, which can hardly be considered "comprehensible" knowledge [3]. Turning to the number of conditions in each rule, the last column of Table 1 shows the value of *Sy*, the simplicity term used in our fitness function. As shown in the table, all the 12 rules have a high value of this measure. As a striking evidence of the simplicity of the rules, they are shown in Appendix A.

CONCLUSIONS AND FUTURE WORK

Despite the small number of examples available in our application domain (taking into account the large number of attributes), the results of our experiments can be considered very promising. The discovered rules had a good performance concerning predictive accuracy, considering both the rule set as a whole and each individual rule. Furthermore, what is more important from a data mining viewpoint, the system discovered some comprehensible rules, as discussed above. It is interesting to note that the system achieved very consistent results by working from "tabula rasa", without any background knowledge, and with a small number of examples.

We should emphasize that our system is still in an experimental, research stage. Therefore, the results presented here should not be used alone for real-world diagnoses without consulting a physician.

Future research would be to perform a careful selection of attributes in a preprocessing step, so as to reduce the number of attributes (and the corresponding search space) given to the GP. Attribute selection is a very active research area in data mining.

Given the results obtained so far, GP has demonstrated to be really useful as a data mining tool, but future work should include also the application of the GP system proposed here to other data sets, to further validate the results reported in this paper.

REFERENCES

1. Assanelli D, Cazzamalli L, Stambini M, *et al*: Correct diagnosis of chest pain by an integrated expert system. *Proc Computers in Cardiology*. IEEE Press: 759-762, 1993.
2. Banzhaf W, Nordin P, Keller RE, Francone FD: *Genetic Programming, an Introduction: on the Automatic Evolution of Computer Programs and its Applications*. Morgan Kauffmann, 1998.
3. Breslow LA, Aha DW: Simplifying decision trees: a survey. *Knowl Engng Rev* 12 (1): 1-40, 1997.
4. Domingos P: Occam's Two Razors: the sharp and the blunt. *Proc. 4th Int Conf Knowledge Discovery & Data Mining*. AAAI Press: 37-43, 1998.
5. Fayyad UM, Piatetsky-Shapiro G, Smyth P: From data mining to knowledge discovery: an overview. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthurusamy R (Eds.): *Advances in Knowledge Discovery & Data Mining*. AAAI/MIT Press: 1-34, 1996.
6. Freitas AA. On objective measures of rule surprisingness. *Principles of Data Mining & Knowledge Discovery: Proc. 2nd European Symp.*, Lecture Notes in Artificial Intelligence 1510, Springer-Verlag: 1-9, 1998.
7. Freitas AA, Lavington SH: *Mining Very Large Databases with Parallel Processing.* Kluwer, 1998.
8. Goldberg D: *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.
9. Hand D: *Construction and Assessment of Classification Rules*. John-Wiley & Sons, 1997.
10. Holland JH: *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
11. Michalewicz Z: *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.
12. Koza JR: *Genetic Programming: on the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.
13. Koza JR: *Genetic Programming II: Autonomous Discovery of Reusable Programs.* MIT Press, 1994.
14. Lopes HS, Coutinho MS, Heinisch R, Barreto JM, Lima WC. A Knowledge-based system for decision support in chest pain diagnosis. *Med Biol Engng Comp* 35 (suppl., part I): 514, 1997.
15. Lopes HS, Coutinho MS, Lima WC: An evolutionary approach to simulate cognitive feedback learning in medical domain. In: Sanchez E, Shibata T, Zadeh LA (Eds), *Genetic Algorithms and Fuzzy Logic Systems*. World Scientific: pp. 193-207, 1997.
16. Mair J, Smidt J, Lechleitner P, Dienstl F, Puschendorf B: A decision tree for the early diagnosis of acute myocardial infarction in non-traumatic chest pain patients at hospital admission. *Chest* 108(6): 1502-1509, 1995.

17. Noda E, Freitas AA, Lopes HS: Discovering interesting prediction rules with a genetic algorithm. To appear in *Proc CEC'99 - Congress on Evolutionary Computation*, Washington, DC, 1999.
18. Pazzani MJ, Mani S and Shankle WR. Beyond concise and colourful: learning intelligible rules. *Proc 3rd Int Conf Knowledge Discovery & Data Mining*. AAAI Press: 235-238, 1997.
19. Rulequest Research. "Data mining and knowledge discovery tools". <http://www.rulequest.com/>, may 1999.
20. Zongker D,Punch B, Rand B: *Lil-gp 1.01 - User´s Manual*. Genetic Algorithms Research and Applications Group (GARAGe), Department of Computer Science, Michigan State University, 1996.

Appendix A

Disclaimer: The classification rules listed here should not be used in clinical diagnosis without consulting a physician. They were obtained from a specific case base, and generalizations to other patients may not be valid.

Rule 1: IF starting factor is emotion AND (the pain lasts no more than seconds OR (the pain begins gradually AND the pain irradiates towards the upper left limb)) THEN disease is stable angina.

Rule 2: IF pain lasts more than 30 minutes AND patient usually takes nitrates THEN disease is unstable angina.

Rule 3: IF starting factor is emotion AND pain is continuous THEN disease is acute myocardial infarction.

Rule 4: IF chest x-ray shows dilatation of the ascending or descending aorta THEN disease is aortic dissection.

Rule 5: IF electrocardiogram voltages are low AND echocardiogram shows some evidence of tamponade THEN disease is cardiac tamponade.

Rule 6: IF history of recent surgery in pelvis or lower limbs THEN disease is pulmonary embolism.

Rule 7: IF chest x-ray shows air in the pleural space THEN disease is pneumothorax.

Rule 8: IF echocardiogram shows pericardial bleeding AND (pain occurs when the patient is lying OR pericardial friction is observed) THEN disease is acute pericarditis

Rule 9: IF a relief factor is feeding OR pain irradiates to the right upper quarter of abdomen THEN disease is peptic ulcer.

Rule 10: IF patient has heartburn AND NOT (history of chronic obstructive pulmonary disease) THEN disease is esophageal pain.

Rule 11: IF (starting factor is physical effort AND (pain is in the upper back region OR starting factor is movement)) AND NOT (patient has dyspnea) THEN disease is musculoskeletal disorder.

Rule 12: IF patient takes drugs for anxiety AND pain character is unspecific AND starting factor is emotion THEN disease is psychogenic chest pain.

Biographies

Celia Cristina Bojarczuk was born in Curitiba, Brazil, in 1972. She has graduated in Computer Engineering in the Pontifical Catholic University of Paraná (PUC-PR), Curitiba, Brazil, in 1996. She is currently graduated student in the area of Artificial Intelligence (Evolutionary Computation) in the Federal Center of Technological Education of Parana (CEFET-PR).

Heitor S. Lopes has born in Ibipora, Brazil, in 1962. He received both the degrees of Electrical Engineer and M.Sc from CEFET-PR (Federal Center of Technological Education of Parana), Curitiba, in 1984 and 1990, respectively. He has got the degree of Doctor in Electrical Engineering in 1996 from the Federal University of Santa Catarina. Since 1987 he has been lecturer at the Department of Electronics of CEFET-PR. In 1997 he has founded the Bioinformatics Laboratory at the CEFET-PR. He is member of IEEE SMC and EMB societies. His current research interests are evolutionary computation, data mining and biomedical engineering.

Alex A. Freitas was born in Belo Horizonte, Brazil, in 1964. He got his B.Sc. in 1989, from FATEC-SP and his M.Sc. in 1993 from Federal University of Sao Carlos, both in the area of Computer Science. He got his Ph.D. in Computer Science from the University of Essex, England, in 1997. He was a visiting lecturer at CEFET-PR, Curitiba, Brazil, from 1997 to 1998. He is currently an associate professor at PUC-PR, Curitiba, Brazil. His main research interests are data mining, knowledge discovery and evolutionary algorithms.