Using Grammar Based Genetic Programming for Data Mining of Medical Knowledge

Po Shun Ngan
Department of Computer Science and Engineering
The Chinese University of Hong Kong, Hong Kong
psngan@cse.cuhk.edu.hk

Man Leung Wong
Department of Computer Science and Engineering
The Chinese University of Hong Kong, Hong Kong
mlwong@cse.cuhk.edu.hk

Kwong Sak Leung
Department of Computer Science and Engineering
The Chinese University of Hong Kong, Hong Kong
ksleung@cse.cuhk.edu.hk

Jack C. Y. Cheng
Department of Orthopaedics and Traumatology
The Chinese University of Hong Kong, Hong Kong
jackcheng@cuhk.edu.hk

ABSTRACT

We have developed a rule learning system that employed Grammar Based Genetic Programming for knowledge discovery from databases. A grammar is used as a template for the rule format. Grammar Based Genetic Programming can guide the evolution and the discovery of meaningful rules. The technique Token Competition is used to achieve the learning of multiple rules simultaneously. We have applied the approach to real-life medical databases. The rules discovered can provide insight into and allow better understanding of the medical domains.

1 Introduction

A database not only stores data but also contains precious knowledge. Many hidden and potentially useful relationships may not be realized by the manual analyses. Moreover, the size of data available now is beyond the capability of human analyses. Thus, the use of computers is necessary. Data mining is a term to describe the automated process of discovering knowledge hidden in data.

Concept learning from data using genetic algorithm has been studied by other researchers. In REGAL (Giordana and Neri 1995), distributed genetic algorithm is used to learn first-order logic concept descriptions. It uses a selection operator, called Universal Suffrage operator, to achieve learning of multi-modal concepts. Another system GABIL (Jong *et al.* 1993) uses a flat string representation and the Pittsburgh's approach: a single individual corresponds to a set of rules. It has an adaptive feature that can adaptively allow or prohibit certain genetic operations for certain individuals. GIL (Janikow 1993) also uses the Pittsburgh's approach and utilizes 14 genetic operators to perform generalization, specialization or other modifications to the rules.

In this paper, the techniques and results of data mining from real-life medical databases are presented. Rules are used as the representation for the discovered knowledge. A rule learning system is designed to capture the special patterns that behave significantly above the average. We have employed Genetic Programming (GP) (Koza 1992, Koza 1994) as the search algorithm. The output is no longer a computer program but a set of rules that can represent the discovered knowledge. Grammar is employed for evolving valid rules.

This paper is organized as follows. Section 2 is an overview of Grammar Based GP. Section 3 describes the approach for rule learning. Section 4 presents the applications of the rule learning system on two real-life medical databases. Section 5 is a conclusion.

2 Grammar Based GP

Grammar Based GP (Wong and Leung 1997, Wong and Leung 1995) is an extension to the original GP. It uses a grammar (Hopcroft and Ullman 1979) to control the structure evolved. The grammar serves as a template for the output. A suitable grammar is designed for the problem. The structure evolved from Grammar Based GP will follow this grammar.

Table 1 is an example of a grammar. The symbols with the italic fonts are the *non-terminals* and the symbols with normal fonts are the *terminals*. A production rule with the form $\alpha \rightarrow \beta$ specifies how a non-terminal is expanded. $\alpha \rightarrow \beta|\gamma$ is a short hand of two production rules $\{\alpha \rightarrow \beta, \alpha \rightarrow \gamma\}$. The *start symbol* is assumed to be the first symbol of the first line.

Grammar Based GP uses a derivation tree as the repre-

Expr → (if *Boolean Real Real*)
Boolean → (*Operator Real Real*)
Boolean → T | F
Operator → = | < | > | <= | >=
Real → var1 | var2 | var3
Real → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Table 1 An example grammar. The symbol `if` returns the second argument if the first argument is true, or else the third argument

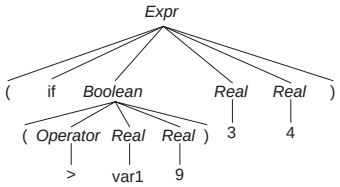

Figure 1 An individual in Grammar Based GP representing the expression (if (> var1 9) 3 4).

sentation of each individual. An individual is created by a complete derivation from the start symbol of the given grammar. Choices are randomly made if there are more than one possible derivation. Figure 1 is an example of an individual derived from the grammar in Table 1.

Similar to GP, there are three genetic operators. *Reproduction* copies one individual into the new population. *Crossover* (Figure 2) and *mutation* (Figure 3) alter the structure in the derivation tree. Crossover produces one child from two parents. One parent is designated as the primary parent and the other is designated as the secondary parent. A subtree of the primary parental derivation tree is selected and replaced by another subtree selected from the secondary parent. However the selection of the replacing subtree is restricted by the grammar. A validation check is made to ensure the new replacing node can match with a production rule of the grammar. Mutation replaces a subtree in the derivation tree by a randomly generated subtree. A node is selected, and the grammar is used to derive a new subtree to replace the subtree on this node. Again, a check is needed to make sure the new tree does not violate the given grammar.

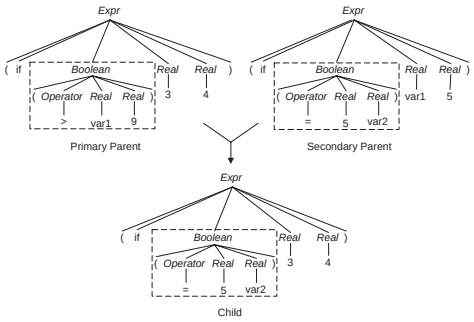

Figure 2 Crossover in Grammar Based GP, evolving the child expression (if (= 5 var2) 3 4).

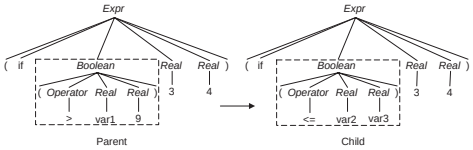

Figure 3 Mutation in Grammar Based GP, evolving the child (if (<= var2 var3) 3 4).

Grammar Based GP is used for rule learning because the grammar can define the format of the rule we want. In pure GP, the function set must satisfy the closure requirement (Koza 1992): Each function must accept, as its arguments, any value and data type returned by any function or assumed by any terminal. Grammar Based GP relaxes this requirement because the placement of a symbol must conform with the grammar. The evolved structure can have different kinds of symbols, and each kind of symbols will only appear in the suitable place.

3 System Design

3.1 System flows

The flowchart of the rule learning system is shown in Figure 4. A grammar is provided as a template for rules. A set of rules is created by using this grammar to make up the initial population. Each rule is evaluated to get a raw fitness score. The evaluation is described in section 3.5. Token competition and replacement, described in section 3.6, aim to retain a set of good individuals and increase the diversity of population. New rules are evolved by three genetic operators, which are detailed in section 3.4. In each generation, an equal number of new individuals are evolved from the original population, but only the best half will be passed to the next generation.

Our system differs from traditional GP by not using reproduction operator, and keeping all parents as competitors in the new generation. Reproduction is not used because we do not want a good rule to replicate itself and dominate the population. Rather, token competition is used in order to find several good rules and diversify the population. On the other hand, we allow them to compete with the offspring, so that good individuals still have chances to live in the next generation.

3.2 Grammar

The grammar can govern the format of the rules to be learned. The format of rules in each problem can be different. The grammar must be written for each problem to best fit the domain. In general, the grammar specifies a format "if [antecedents] then [consequent]". The antecedent part is a conjunction of descriptors. The consequent part is a descriptor. A descriptor is a description on the attribute value. It can specify a value for a nominal attribute, a range for a continuous attribute, or a comparison with other attributes.

As an example, consider a database with 4 attributes.

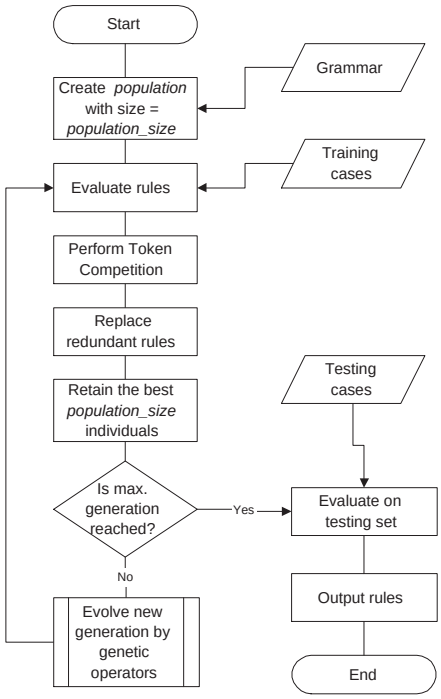

Figure 4 The system flowchart

Rule → if *Antes* , then *Consq* .
Antes → *Attr1* and *Attr2* and *Attr3*
Attr1 → any | *Attr1_descriptor*
Attr2 → any | *Attr2_descriptor*
Attr3 → any | *Attr3_descriptor*
Attr1_descriptor → attr1 = erc1
Attr2_descriptor → attr2 between erc2 erc2
Attr3_descriptor → attr3 *Comparator Attr3_term*
Comparator → = | != | <= | >= | < | >
Attr3_term → attr2 | erc3
Consq → *Attr4_descriptor*
Attr4_descriptor → attr4 = boolean_erc

Table 2 An example grammar for rule learning.

We want to learn rules about **attr4**, which is boolean. The attribute **attr1** is nominal and encoded with 0, 1 or 2. The attribute **attr2** is continuous between 0-200. The domain of **attr3** is similar to **attr2** and we want the rule to compare them. Thus the context free grammar in Table 2 can be used. The symbols **erc1**, **erc2**, **erc3** and **boolean_erc** are ephemeral random constants (ERCs). Each ERC has it own range for instantiation: **erc1** is within {0,1,2}, **erc2** and **erc3** is between 0-200, **boolean_erc** can only be T or F. The symbol '**any**' serves as a wild-card in the rule. An attribute will not be considered in the rule if its attribute descriptor is '**any**'. This grammar allows rules like:

- if **attr1** = 0 and **attr2** between 100 150 and **attr3** != 50, then **attr4** = T.
- if **attr1** = 1 and any and **attr3** >= **attr2**, then **attr4** = F.

The grammar is used for creating individuals. The start symbol is **Rule** and a complete derivation is performed. Then the ERCs are instantiated. These constants can be instantiated randomly, or by using *seeds*. If seeds are used, the constants are assigned values from a random record in the training set.

3.3 Temporal Order and Causality

The use of grammar can ensure syntactical correctness in the rule, but not semantical correctness. For example, in a medical domain, the rule "if treatment is plaster, then diagnosis is Radius fracture" is inappropriate. This rule does not make sense, because an operation is taken based on the treatment, not the other way round. This rule does not provide any hint on the cause of treatment. It is desirable to eliminate meaningless cause in the search process. This requires a certain degree of knowledge on the causality between the attributes. However, this knowledge may not be readily available. A simple way is to order the attributes according to the temporal relationship. An event that occurs later will not be a cause of an event occurred earlier! This information can be incorporated in the design of the grammar. An attribute should not be place in the 'if' part if it occurs later then the attribute in the 'then' part. This causality model is easy to construct and applicable to all problems. The search space can be reduced drastically and meaningless rules can be eliminated.

3.4 Genetic Operators

New rules are evolved by using three genetic operators: crossover, mutation and dropping condition. These operators modify the structure of the derivation tree and thus change the attribute descriptors of the rules. Parents for these operators are selected using rank selection (Goldberg 1989).

Crossover replaces a subtree of a primary parental derivation tree by a subtree from a secondary parental derivation tree. Crossover may occur at a random point in the derivation tree, but the grammar can maintain the validity of the rule. A conjunction of descriptors can only crossover with another conjunction of descriptors. A single descriptor can only crossover with another descriptor of the same attribute. Mutation re-creates a subtree of the derivation tree using the grammar. Mutation may also occur at a random point. It can mutate the whole rule, an attribute descriptor or just an ERC in the rule.

Due to the probabilistic nature of GP, redundant constraints may be generated in the rule. For example, suppose the actual knowledge is 'if A<20 then X=T'. We may learn rules like 'if A<20 and B<20 then X=T'. This rule is correct but does not completely represent the actual knowledge. The rule can be generalized if one descriptor in the antecedent part is dropped. Dropping condition selects randomly one attribute descriptor, and then turn it into 'any'. Thus that particular attribute is

no longer considered in the rule.

3.5 Evaluation of Rules

The support-confidence framework (Agrawal *et al.* 1993) is employed as the evaluation measure. *Support* measures the coverage of a rule while *confidence factor* measures the consistency of a rule.

The data set should be partitioned into a training set and a testing set. In the evaluation process, each rule is checked with every record in the training set. The discovered rules after the learning process are further evaluated with the unseen testing set, so as to verify their accuracy.

The *confidence factor* (cf) is the ratio of the number of records matching both the antecedents and the consequence to the number of records matching only the antecedents. But a rule with a high confidence factor does not imply it behaves significantly different from the average. We need to compare with the average probability ($prob$) of consequent, which is the occurrence of the consequent over total number of records in the training set. This value measures the confidence for the consequence under no particular antecedent.

We defined cf_part as

$$cf\_part = 2 \times cf \times \log(\frac{cf}{prob})$$

This value is based on two factors : cf and $cf/prob$. The log function measures the order of magnitude of the ratio $cf/prob$. A high value of cf_part requires the rule to have a high accuracy (cf) and cf is higher than the average probability ($prob$).

A rule can have a high accuracy but the rule may be just because of chance and based on a few training examples. This kind of rule does not have enough support. The value *support* is the ratio of the number of records covered by the rule to the total number of records. If *support* is below a minimum required support, min_supp, the confidence factor of the rule should not be considered.

We define our fitness function to be :

$$\begin{aligned} & raw\_fitness \\ = \ & support, \text{ if } support < min\_supp \\ = \ & w_1 \times support + w_2 \times cf\_part + w_3, \text{ otherwise} \end{aligned}$$

where the weights w_1, w_2 and w_3 are user-defined. By tuning these weights, the user can control the balance between the confidence and support. The values we used are 1, 8 and 0.03 respectively. The value of min_supp is set to 3%.

3.6 Token Competition

A rule learning system should be able to learn as many interesting rules as possible. Thus the problem should be modeled as the searching of multiple peaks in the search space. We follow the Michigan approach (Holland and Reitman 1978, Booker *et al.* 1989): Each individual corresponds to one rule, and the population provides a rule set to represent the knowledge. The technique token competition (Leung *et al.* 1992) is employed to achieve the *niching* effect, so that good individuals in different niches are maintained in the population. Token competition does not need to define and calculate the similarity between individuals. It simply regards two individuals to be similar if they cover the same record.

In the natural environment, once an individual has found a good place for living, it will try to exploit this niche and prevent other newcomers to share the resources, unless the newcomer is stronger. The other individuals are hence forced to explore and find their own niches. In this way, the diversity of the search population is increased.

Based on this mechanism, we assume each record in the training set can provide resources called tokens. If a rule can match a record, it will seize the token and other weaker rules cannot get it. The priority of receiving tokens is determined by the strength of the rules. A rule with a high fitness score can exploit the niche by seizing as many tokens as it can. The other rules entering the same niche will have their strength decreased because it cannot compete with the stronger rule. We defined the modified fitness as :

$$modified\_fitness = raw\_fitness \times count/ideal$$

where *count* is the number of tokens that the rule actually seized, *ideal* is the ideal number of tokens that it can seize, which is equal to the number of records that the rule matches, $raw\_fitness$ is the fitness score obtained from the evaluation process.

From another point of view, each rule contributes to the system by covering a portion of significant records of the database. If a record has been covered by one rule, then another rule covering the same record will make no contribution to the system. Thus the fitness of this rule should be discounted.

As a result of token competition, there will be rules that cannot seize any token. These rules are redundant as all of its records are already covered by stronger rules. They can be replaced by new individuals so as to inject a larger degree of diversity into the population.

4 Application on Real-life Medical Databases

The rule learning system has been applied to two real-life medical databases. These databases are obtained from the Orthopaedic Department, Prince Wales Hospital of Hong Kong.

4.1 The fracture database

The first medical database contains admission records of children with fractures admitted to the hospital. This database has 6500 records and 7 attributes. The attributes in the database are sex, age, admission date, diagnosis, operation, surgeon of operation and length of

	No. of Rules	*cf*			*cf/prob*			*support*		
		mean	max	min	mean	max	min	mean	max	min
Diagnosis	2	45.56%	51.43%	39.75%	1.56	1.70	1.42	9.22%	10.01%	8.42%
Operation	8	42.64%	74.05%	27.96%	1.99	2.86	1.14	5.38%	16.22%	3.19%
Length of stay	7	71.11%	81.11%	46.98%	2.51	6.92	1.39	4.46%	8.65%	3.09%

Table 3 Summary of the rules for the fracture database

staying in hospital. These data can provide information for the analysis of children fracture patterns.

We can divide the attributes into three causality models. Firstly, sex, age and admission date are the possible causes of diagnosis. Secondly, the operation is taken after the diagnosis. The possible causes of operation and surgeon are sex, age, admission date and diagnosis. Thirdly, length of staying is the last event and has the other 6 attributes as the possible causes. This knowledge follows directly from the temporal relationships.

A grammar is written as a template for these three kinds of rules. A set of interesting rules is learned from the database using our rule learning system, and the result is summarized in Table 3.

From the rules about diagnosis, we found that humerus fracture is the most common fracture for children between 2 and 5 years old. Radius fracture is the most common fracture for boys between 11 and 13. The confidence factor is not high because the database did not have related attributes that strongly affect the value of diagnosis. However the ratio *cf/prob* shows that the patterns discovered deviated significantly from the average.

The rules about operation/surgeon suggest that radius and ulna fractures are usually treated with plaster. Operation is usually not need for tibia fracture. Open reductions are more common for elder children with age larger than 11, while young children with age lower than 7 have a higher chance of not requiring operations. We did not find any interesting rules about surgeons.

The rules about length of stay suggest that Femur and Tibia fractures are serious injuries and have to stay longer in hospital. If open reduction is used, the patient requires longer time to recover because the wound has been cut open for operation. If no operation is needed, it is likely that the patient can return home within one day. Relatively, radius fracture requires a shorter time for recovery.

4.2 The Scoliosis database

The second database contains clinical records of patients with Scoliosis. Scoliosis refers to the deformation of the backbone. The database records measurements on the patients, such as the number of curves, the curve locations, degrees and directions. It also records maturity of the patient, class of Scoliosis and treatment. The database has 500 records and 15 attributes.

There can be two types of rules mined from this domain. The first type is rules for classification of Scoliosis. Scoliosis can be classified as Kings, Thoracolumbar(TL) and Lumber(L), while Kings can be further subdivided into King-I, II, III, IV and V. According to the domain knowledge, the classification mainly depends on the locations and degrees of the curves. The second type of rule is for suggesting treatment. Treatment can be observation, surgery and bracing. According to the domain knowledge, treatment mainly depends on age, laxity, degrees of the curves, maturity of the patient, displacement of the vertebra and the class of Scoliosis. As Grammar Based GP is used, the domain knowledge can be easily incorporated in the design of the rule grammar.

For each class of Scoliosis, a number of rules are mined. The results are summarized in Table 4. These results have been compared with the knowledge of doctors. For King-I and II, the rules have high confidence and generally match with the domain knowledge. However there is one unexpected rule for the classification of King-II. After an analysis on the database, our rule revealed that serious data errors existed in the current database.

For King-III and IV, the confidence of the rules discovered is fairly low. According to the domain knowledge, these two classes should have only one major curve. However the concept of 'major curve' is fuzzy and cannot be deduced from just the degrees. Without this important information, the system cannot find accurate rules for these two classes .

For the class TL and L, the infrequent occurrences of these two classes affects the learning performance. Nevertheless, the rules show something different with the existing domain knowledge. According to our rules, the classification depends on the *first major curve*, while according to the domain knowledge, the classification depends on the *larger major curve*. After discussion with the domain expert, it is agreed that our rules are more accurate than the existing domain knowledge.

The results of rules about treatment are summarized in Table 5. The rules for observation and bracing have high confidence factors, with a trade-off of lower supports in the weights setting. For surgery, we failed to find any interesting rules because surgery only occurred in 3.65% of the database.

5 Conclusion

We have presented a rule learning system for discovery knowledge from databases. Our approach has employed several new techniques to tackle the task. Grammar Based GP has been employed to enhance the evolutionary

Class	No. of Rules	*cf*			*prob*	*support*		
		mean	max	min		mean	max	min
King-I	5	94.84%	100%	90.48%	28.33%	5.67%	10.73%	0.86%
King-II	5	80.93%	100%	52.17%	35.41%	6.61%	14.38%	1.07%
King-III	4	23.58%	25.87%	16.90%	7.94%	1.56%	2.58%	0.86%
King-IV	2	24.38%	29.41%	19.35%	2.79%	1.18%	1.29%	1.07%
King-V	5	54.13%	62.50%	45.45%	6.44%	0.97%	1.07%	0.86%
TL	1	41.18%	41.18%	41.18%	2.15%	1.50%	1.50%	1.50%
L	3	54.04%	62.50%	45.45%	4.51%	2.00%	2.79%	1.07%

Table 4 Summary of the rules for classification of Scoliosis

Type	No. of Rules	*cf*			*prob*	*support*		
		mean	max	min		mean	max	min
Observation	4	98.89%	100%	95.55%	62.45%	3.49%	6.01%	1.07%
Bracing	5	79.57%	100%	71.43%	24.46%	1.03%	1.29%	0.86%
Surgery	0	-	-	-	3.65%	-	-	-

Table 5 Summary of the rules about treatment

process. The grammar can ensure proper placement of symbols in crossover and mutations, and thus can maintain the rule format. Token competition has been used for learning multiple rules. This simple approach can force the individuals to explore the search space and achieve the niching effect.

Data mining can discover new knowledge as well as refine our existing knowledge. The system has been applied to two real-life medical databases. In the first database, we can automatically uncover the relationships among the variables. In the second database, we have discovered unexpected rules that disagree with the existing domain knowledge. After an analysis we have found out data errors in the database. On the other hand, the existing knowledge has been refined based on the new discovered knowledge.

6 Acknowledgments

The work was partially supported by XXXXX XXXX XXXXXX XXXXXX XXXXXX XXXXXX. The authors wish to thank Ms. Chun Sau Lau and Ms. King Sau Lee for preparing, analyzing and implementing the rule learning system for the Scoliosis database.

References

Agrawal, R., T. Imielinski and A. Swami (1993). Mining association rules between sets of items in large databases. In: *Proceedings of the 1993 International Conference on Management of Data (SIGMOD 93)*. pp. 207–216.

Booker, L., D. E. Goldberg and J. H. Holland (1989). Classifier systems and genetic algorithms. *Artificial Intelligence* **40**, 235–282.

Giordana, A. and F. Neri (1995). Search-intensive concept induction. *Evolutionary Computation* **3**, 375–416.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Holland, J. H. and J. S. Reitman (1978). Cognitive systems based on adaptive algorithms. In: *Pattern-Directed Inference Systems* (D. A. Waterman and F. Hayes-Roth, Eds.). Academic Press.

Hopcroft, J. E. and J. D. Ullman (1979). *Introduction to automata theory, languages, and computation.*. Reading, MA: Addison-Wesley.

Janikow, C. Z. (1993). A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning* **13**, 189–228.

Jong, K. A. De, W. M. Spaers and D. F. Gordon (1993). Using genetic algorithms for concept learning. *Machine Learning* **13**, 161–188.

Koza, J. R. (1992). *Genetic Programming : on the programming of computers by means of natural selection*. Bradford/MIT Press.

Koza, J. R. (1994). *Genetic Programming II : automatic discovery of reusable programs*. Bradford/MIT Press.

Leung, K. S., Y. Leung, L. So and K. F. Yam (1992). Rule learning in expert systems using genetic algorithm: 1, concepts. In: *Proceedings of the 2nd International Conference on Fuzzy Logic and Neural Networks(Iizuka, Japan)*. pp. 201–204.

Wong, M. L. and K. S. Leung (1995). Inducing logic programs with genetic algorithms: The genetic logic programming system. *IEEE Expert* **10**(5), 68–76.

Wong, M. L. and K. S. Leung (1997). Evolutionary program induction directed by logic grammars. *Evolutionary Computation* **5**, 143–180.