2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Learning non-overlapping rules A method based on Functional Dependency Network and MDL Genetic Programming

Wing-Ho Shum, Kwong-Sak Leung
Dept. of Computer Science & Engineering
The Chinese University of Hong Kong
Hong Kong
{whshum, ksleung}@cse.cuhk.edu.hk

Man-Leung Wong
Dept. of Computing & Decision Sciences
Lingnan University
Hong Kong
mlwong@ln.edu.hk

Abstract— **Classification rule is a useful model in data mining. Given variable values, rules classify data items into different classes. Different rule learning algorithms are proposed, like Genetic Algorithm (GA) and Genetic Programming (GP). Rules can also be extracted from Bayesian Network (BN) and decision trees. However, all of them have disadvantages and may fail to get the best results. Both of GA and GP cannot handle cooperation among rules and thus, the learnt rules are likely to have many overlappings, i.e. more than one rules classify the same data items and different rules have different predictions. The conflicts among the rules reduce their understandability and increase their usage difficulty for expert systems. In contrast, rules extracted from BN and decision trees have no overlapping in nature. But BN can handle discrete values only and cannot represent higher-order relationships among variables. Moreover, the search space for decision tree learning is huge and thus, it is difficult to reach the global optimum. In this paper, we propose to use Functional Dependency Network (FDN) and MDL Genetic Programming (MDLGP) to learn a set of non-overlapping classification rules [17]. The FDN is an extension of BN; it can handle all kind of values; it can represent higher-order relationships among variables; and its learning search space is smaller than decision trees'. The experimental results demonstrate that the proposed method can successfully discover the target rules, which have no overlapping and have the highest classification accuracies.**

I. Introduction

Classification rules classify data items according to their variable values. In the other words, rules represent relationships among variables and classes. Many rule learning algorithms are proposed, like GA and GP and rules can also be extracted from BN and decision trees. However, all of them have disadvantages and may fail to learn the best rules.

GA and GP have been applied to different areas, like shortest path finding and classification [2]. While compared with GA, GP has the greater representation power which can represent complicated solutions, like classification tree and Prolog statement [11], [19].

There are two ways to represent rules in GA and GP, the Pittsburgh and the Michigan approaches [7], [6]. With the first approach, each individual in the population represents a set of rules, so the search space is huge and it is difficult to reach the global optimum. With the latter, each individual in the population represents a single rule, i.e. a part of a candidate solution. As individuals evolve and be evaluated separately, there is no cooperation among them. In result, the learnt rules are likely to have many overlappings, i.e. the conflicts among the rules.

Typically, the conflicts among the rules can be solved by, 1) sorting the rules according to their fitness, higher classification accuracy higher order, i.e. the first one is the best; 2) checking the data items if they satisfy the rules' conditions; 3) if a data item satisfies more than one rules, only the one with the highest order would be chosen; 4) firing the chosen rules to predict the classes. Although the method works well for classification, it is difficult to realize rules' effective meaning.

For example, if we want to know the effective meaning of the sixth rule, we need to combine it with the neglects of the preceding ones. It would be even worsen if we want to merge the rules into an expert system, as there may have couples or even hundreds of rules. Therefore, rules with overlappings have the lower understandability and the higher usage difficultly for expert systems.

Figures 1 and 2 illustrate the synthetic data set used in [16] and the data set with the first eight rules' coverages respectively. The data set has 2 variables, 4 classes and 4000 data items and the rules were learnt by a canonical GP. A rule's coverages are the parts of the problem space satisfying the conditions. From the figures, we can see that there are many overlappings among different rules' coverages. For instance, with the typical method, the eighth rule would possibly be fired only if the data items cannot satisfy all of the preceding ones. In the other words, to interpret the eighth rule's effective meaning, we need, where it is not always possible to combine it with the neglects of the others.

Table I shows the first six rules for the same data set in another run. It is hard to combine the sixth rule with the neglects of the preceding ones.

BN and decision trees can produce rules without overlapping in nature. But BN can only handle discrete values, continuous, interval and ordinal ones must be discretized and thus, the order information is lost [13]. The disability to handle continuous, interval and ordinal values increases the network complexity; the relationships represented by BN

TABLE I

THE FIRST SIX RULES FOR THE SYNTHETIC DATA SET.

1. if (var_1 between 0.14 and 1.46321) and (var_0>0.19) then (class=4)
2. if (var_1>0.524064) and (var_0 between 0.38 and var_1) and (var_0 between 0.524064 and 0.59) then (class=2)
3. if (var_0 between 0.524064 and var_1) then (class=3)
4. if (var_1>var_0) and (var_1 between 0.14 and 0.37) then (class=1)
5. if (var_0 between 0.524064 and 0.01) then (class=3)
6. if (var_1≤0.524064) and (var_1>var_0) and (var_0 between 0.524064 and 0.59) then (class=2)

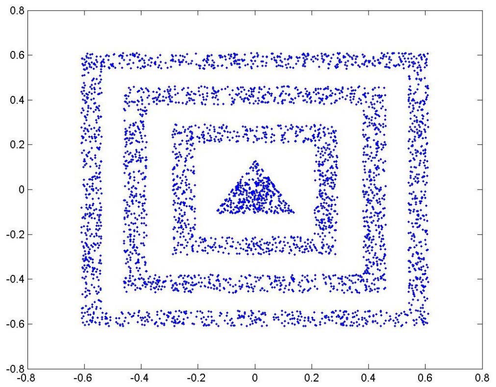


Fig. 1. The synthetic data set.

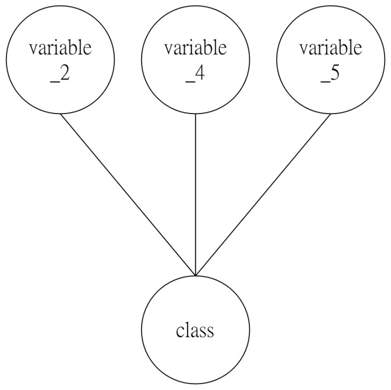


Fig. 4. The best Bayesian Network for Monk 3.

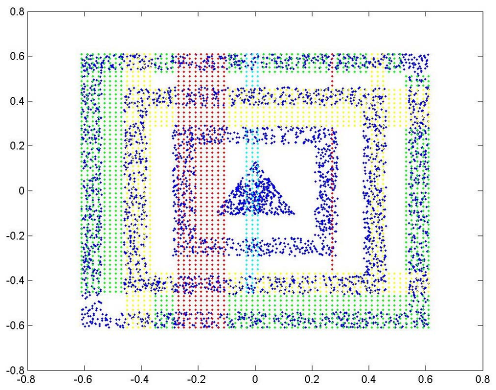


Fig. 2. The synthetic data set with the first eight rules' coverages.

become less understandable; and higher-order relationships among variables cannot be discovered.

Decision tree is a powerful classification model. Well-known decision tree learning algorithms include classification and regression trees (CART), OC1, Ltree, C4.5 and C5.0 [1], [12], [14], [3], [9]. However, as they build decision trees step by step and node by node, they may get stuck into local optima. A step looks good in an earlier stage may finally turn out to be bad. Weak methods, like GA and GP are then proposed to learn decision trees [11], but have huge search space. In a BN, each of the variables can appear once at most, but they can appear many times in the decision tree counterpart. Therefore, even GA and GP are hard to reach the global optimum.

Figures 3 and 4 show the best decision tree and the best BN for the benchmark data set, Monk 3 respectively [5]. The decision tree is more complicated and has more nodes than its BN counterpart.

So all of GA, GP, BN and decision trees are hard to get the best results. We propose to use the FDN and the MDLGP to learn a set of non-overlapping classification rules [17].

The FDN is an extension of BN, it can handle all of discrete, continuous, interval and ordinal values and can also represent higher-order relationships among variables. Like BN, the rules extracted from BN have no overlapping in nature.

Consider a university programme selection problem that has two ordinal variables. Suppose a high school student would choose a science programme if he/she got a better grade in biology; he/she would study an art programme if he/she done has better in history; otherwise, he/she would study one randomly. Figure 6 shows the BN representing the problem. Since BN cannot compare the course grades directly, it enumerates all the instances of the combination of the subject grades and calculates the corresponding probabilities. Although there are only two subjects, the conditional tables are large and have a lot of entries, i.e. the network complexity is high and the meanings of the relationships are unclear and incomprehensible. The BN shows there are some relationships, but it cannot illustrate that it is the grade comparison result affects the programme selection. The understandability of the relationships is reduced.

Figure 5 shows the FDN for the programme selection problem. The FDN has a functional node, $history > biology$. It represents the grade comparison between the two subjects and its conditional table has only one entry specifying the

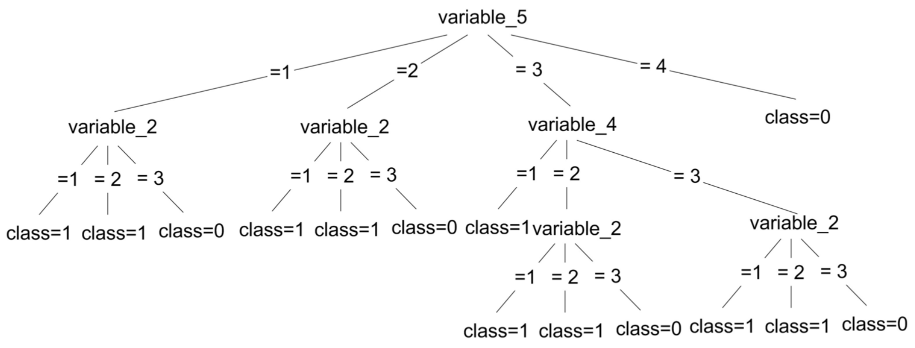
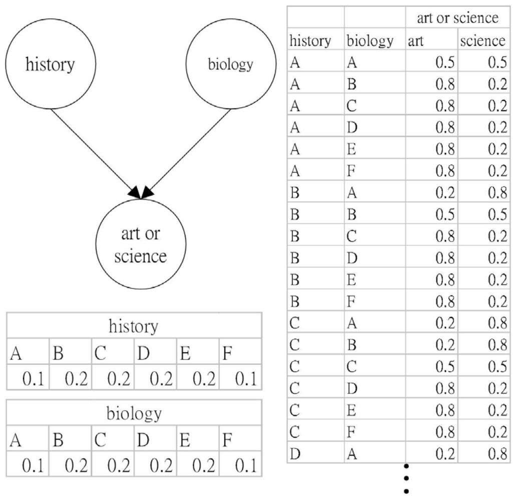


Fig. 3. The best decision tree for Monk 3.



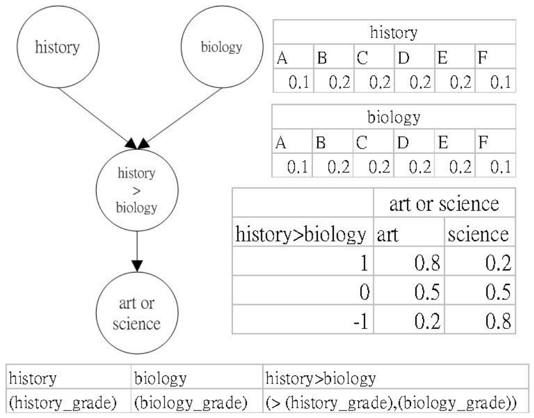
history					
A	B	C	D	E	F
0.1	0.2	0.2	0.2	0.2	0.1

biology					
A	B	C	D	E	F
0.1	0.2	0.2	0.2	0.2	0.1

		art or science	
history	biology	art	science
A	A	0.5	0.5
A	B	0.8	0.2
A	C	0.8	0.2
A	D	0.8	0.2
A	E	0.8	0.2
A	F	0.8	0.2
B	A	0.2	0.8
B	B	0.5	0.5
B	C	0.8	0.2
B	D	0.8	0.2
B	E	0.8	0.2
B	F	0.8	0.2
C	A	0.2	0.8
C	B	0.2	0.8
C	C	0.5	0.5
C	D	0.8	0.2
C	E	0.8	0.2
C	F	0.8	0.2
D	A	0.2	0.8

⋮

Fig. 5. The Bayesian Network for the programme selection problem.



history					
A	B	C	D	E	F
0.1	0.2	0.2	0.2	0.2	0.1

biology					
A	B	C	D	E	F
0.1	0.2	0.2	0.2	0.2	0.1

	art or science	
history>biology	art	science
1	0.8	0.2
0	0.5	0.5
-1	0.2	0.8

history	biology	history>biology
(history_grade)	(biology_grade)	(> (history_grade),(biology_grade))

Fig. 6. The Functional Dependency Network for the programme selection problem.

function $>$. $>$ returns 1 if the first argument is greater than the second one; if the two arguments are equal, it returns 0; otherwise, -1 is returned. With the functional node, the FDN has reduced the number of entries in the conditional tables from $6 + 6 + 6^2 * 2$, i.e. 84 to $6 + 6 + 3 * 2 + 1$, i.e. 19, the network complexity is significantly reduced. By realizing the meaning of $>$, it is easy to understand and interpret the meanings of the relationships; and the relationships can be expressed in rule format easily. Furthermore, the number of rules is proportional to the number of entries in the conditional tables, so the FDN can produce a smaller rule set. The smaller number of rules for expert systems the more savings on computation resources.

The MDLGP is a variant of GP proposed to learn the FDN, which uses an extended MDL to evaluate the candidate solutions. It does not employ any knowledge-guided nor application-oriented operator, thus it is robust and easy to be replicated.

We also propose a procedure to extract classification rules from the FDN. The paper is organized as follows. We introduce the FDN in Section II-A, followed by a description of the MDLGP and the rule extraction procedure. The experimental results are presented in Section III. A conclusion is given in the last section.

II. The Algorithm

A. The Functional Dependency Network

The FDN is a directed acyclic network. Figure 6 shows an example of the FDN. A variable node denotes a variable in the domain and a directed link, $N_i \rightarrow N_j$ represents the dependency relationships between the child N_j and the parent N_i. Each variable node has a conditional table specifying the probability of each particular value of the variable node given an instantiation of the parents. In other words, for each variable node N_i with parents Υ_{N_i}, there is a conditional table specifying the conditional probability distribution $P(N_i|\Upsilon_{N_i})$; for each N_i with no parent, the conditional table specifying the priori probability distribution $P(N_i)$.

The FDN also has one more type of nodes, functional node which represents functions of variables. The functions can have any number of arguments and any number of nesting levels. Their conditional tables have only one entry, which specifies the functions producing the value of the functional node given an instantiation of its parents.

Similar to BN, a variable node can handle discrete values only. If a continuous functional node has a variable node as its child, discretization is needed.

A continuous variable node can either have another variable node or a continuous functional node or both as its children. If the child is a variable node, discretized values are produced; if the child is a continuous functional node, continuous values are generated according to a Gaussian distribution function. Each entry in the conditional table represents an interval. To generate a continuous value, 1) select an entry according to the probability, 2) according to Gaussian distribution function, pick up a continuous value randomly given the mean and the standard deviation of the interval.

B. The MDL Genetic Programming

1) The Population: The MDLGP is designed to learn the FDN. It has a population of individuals. Individuals are represented as trees and each individual encodes one network. Individuals are of the form, $(< parents > \rightarrow < child >)_1 .....(< parents > \rightarrow < child >)_y$ where $y \in Z+$. We call each of the $(< parents > \rightarrow < child >)$ as a fragment, which represents the relationships between the $< parents >$ and the $< child >$. $< parents >$ and $< child >$ are in the prefix form. $< parents >$ denotes one or more parents and a parent can either be a variable node or a functional node. $< child >$ is a variable node. Different fragments can have the same $< child >$. The fragments containing functional nodes also represent the variables in the functions. For instance, the individual representing the FDN in Figure 6 is $((> \textit{history biology}) \rightarrow \textit{art\_or\_science})$

The MDLGP uses a grammar to prevent the closure problem, i.e. the type mismatching issue among functions and variables. Table II shows the grammar.

The MDLGP translates an individual into a network fragment by fragment. Individuals may carry invalid fragments, which would create cycles in the network. The MDLGP validates them one by one, starting from the leftmost one. If the fragment is valid, it would be translated as links and nodes into the network. If it would create cycle in the network, it would be simply ignored.

The population has a fixed number of individuals and the initial population is generated randomly.

2) Extended MDL Calculation: The MDLGP uses an extended MDL to evaluate the fitness of individuals. MDL balances between the network's accuracy and the simplicity; and it is a combination of network description length and data description length. The smaller MDL score is the better.

Let G be a network, $U = \{N_1, ..., N_n\}$ be the variables in the domain, $V = \{F_1, ..., F_f\}$ be the functional nodes in G, Υ_ϕ be the parents of the variable node or the functional node ϕ, s_ϕ be the number of possible states of the variable node or the functional node ϕ and d is the number of bits required to store a numerical value. The MDL score of G is the sum of the MDL scores of U and V.

$$MDL(G) = \sum_i MDL(N_i, \Upsilon_{N_i}) + \sum_j MDL(F_j, \Upsilon_{F_j}) \tag{1}$$

a variable node or a functional node ϕ's MDL score is given by

$$MDL(\phi, \Upsilon_\phi) = ND(\phi, \Upsilon_\phi) + DD(\phi, \Upsilon_\phi) \tag{2}$$

where ϕ is either N_i or F_j

The network description length measures the number of bits encoding the network. To represent a particular FDN, the following information is necessary and sufficient:

• A list of the parents of each variable node and the number of bits required is $|\Upsilon_{N_i}| log_2(n + f)$.

• The set of conditional probabilities associated with each variable node and the number of bits required is $d(s_{N_i} - 1) \prod_{\varphi \in \Upsilon_{N_i}} s_\varphi$.

• The functional description associated with each functional node. For instance, the functional description of the functional node $(> \textit{variable\_1 variable\_3})$ is $> v1v3$ and the length is $5 * 8$, i.e. 40 bits.

The network description length of a node, $ND(\phi, \Upsilon_\phi)$ is

$$ND(\phi, \Upsilon_\phi) = \{ \begin{array}{ll} fdl(\phi) & \text{if } \phi \text{ is } F_j \\ |\Upsilon_\phi| log_2(n + f) + d(s_\phi - 1) \prod_{\varphi \in \Upsilon_\phi} s_\varphi & \text{else} \end{array} \tag{3}$$

where $fdl()$ is the length of the functional description and is measured in term of bits

The data description length measures the number of bits encoding the data set given the network, using Huffman code and the probabilities of occurrences of each instantiation in the data set. Closer the probabilities represented by the network to the correct ones smaller the data description length. Functional nodes' data description lengths are counted as 0, because 1) they simply apply the functions on the parents' instantiations and the relationships are absolutely certain (according to the MDL principle, the data description length would be calculated as 0 if the relationship is certain); 2) the calculation results are not actually encoded in the data set.

The data description length of a node, $DD()$ is

$$DD(\phi, \Upsilon_\phi) = \{ \begin{array}{ll} 0 & \text{if } \phi \text{ is } F_j \\ \sum_{\phi, \Upsilon_\phi} M(\phi, \Upsilon_\phi) log_2 \frac{M(\Upsilon_\phi)}{M(\phi, \Upsilon_\phi)} & \text{else if } \Upsilon_\phi \neq \{\} \\ \sum_{\phi, \Upsilon_\phi} M(\phi) log_2 \frac{e}{M(\phi)} & \text{else} \end{array} \tag{4}$$

where $M()$ is the number of data items that match a particular instantiation in the data set and e is the total number of data items in the data set (the log_2 function will be 0 if $M(\Upsilon_\phi)$ is 0).

3) Selection and Reproduction: The MDLGP selects individuals for reproduction through tournament competition. Each individual competes with a number of randomly chosen individuals. According to the number of winning competitions, fitter individuals are selected. The MDLGP selects individuals by the Roulette Wheel method [10].

The MDLGP has four genetic operators, they are the mutation, the crossover, the insertion and the deletion. The mutation and the crossover are the canonical ones of GP

TABLE II

THE GRAMMAR FOR THE MDL GENETIC PROGRAMMING.

```
network := link*
link := (parents → child)
parents := parent*
parent : = node || functional_node
child := any_variable
node := any_variable
functional_node := discrete_fnode || continuous_fnode || ordinal_fnode|| interval_fnode
discrete_fnode := d_comparison
continuous_fnode := c_comparison || c_operation
ordinal_fnode := o_comparison
interval_fnode := i_operation
d_comparison := (> discrete_variable discrete_variable)
c_comparison := (> continuous_variable continuous_variable)||
                (> c_operation continuous_variable)||
                (> c_operation c_operation)
o_comparison := (> ordinal_variable ordinal_variable)
c_operation := (coperator continuous_variable continuous_variable)
i_operation := (ioperator interval_variable interval_variable)
coperator := + || - || * || /
ioperator := + || -

where > returns 1 if the first argument is larger than the second one; it returns 0 if
they are equal; else it returns -1 and * denotes one or more occurrences
```

[8].The insertion inserts a randomly generated fragment or a parent into a random position of the selected individual and the deletion deletes a randomly chosen fragment or a parent from the selected individual. All the offsprings have to conform the Grammar.

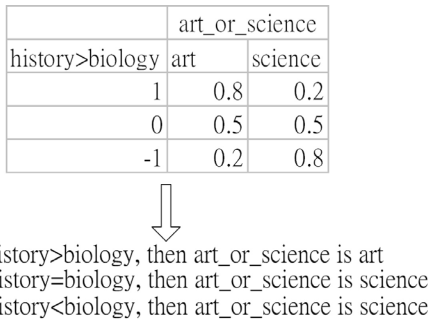
After the reproduction, the total number of individuals and offspring is double. To keep the population size remain constant, the worst half of them are destroyed. Then, the extinction is used to further promote the diversity of the population [4].

C. Rule Extraction

Once the maximum number of generations is met, the learning process is stopped and the fittest individual in the population is chosen as the final solution. Then, classification rules are extracted from the final solution's conditional tables.

The FDN is an extension of BN and classifies data items with probabilities. To generate classification rules, the probabilities are removed and each of the rows in the conditional tables is translated to be a rule. As different rows represent different instantiations of the parents, the translated rules must have no overlapping. For each of the rows in the conditional tables, 1) with the meanings of the functions, translate the instantiation of the parents to be the conditions of the rule; 2) among the class values, select the one with the highest probability to be the prediction; and 3) if more than one class values have the highest probability, select one randomly.

Figure 7 shows an example of the rule extraction. There are 3 rules. The first and the third ones simply use the class values with the highest probability as the predictions. In the case of the second one, the two class values have the same probability, thus either one, i.e. science is chosen.



	art_or_science	
history>biology	art	science
1	0.8	0.2
0	0.5	0.5
-1	0.2	0.8

1. if history>biology, then art_or_science is art
2. if history=biology, then art_or_science is science
3. if history<biology, then art_or_science is science

Fig. 7. An example of the rule extraction.

III. EXPERIMENTAL RESULTS

A. Experimental setting

1) Data Sets: We have evaluated the rules learnt by the FDN on three real-life data sets from UCI machine learning repository [5]. Data set 1 is the Monk 3 data set. It has 7 nominal variables and 556 data items. The variables are $variable\_0$, $variable\_1$, $variable\_2$, $variable\_3$, $variable\_4$, $variable\_5$ and $variable\_6$. The values of $variable\_2$, $variable\_4$ and $variable\_5$ determine the values of $variable\_6$. It can also be classified correctly by the decision tree shown in figure 3.

Data set 2 is a data set that models the psychological experiments reported by Siegler [18]. It has 5 continuous variables and 625 data items. Each data item is classified as having the balance scale tip to the right, tip to the left or be balanced. The variables are $left\_distance$, $left\_weight$, $right\_distance$ and $right\_weight$. The correct way to find the class is the greater of $(left\_distance * left\_weight)$

and $(right\_distance * right\_weight)$. If they are equal, it is balanced.

Data set 3 is the Monk 1. It has 7 nominal variables and 556 data items. The variables are $variable\_0$, $variable\_1$, $variable\_2$, $variable\_3$, $variable\_4$, $variable\_5$ and $variable\_6$. The relationships, $variable\_1 = variable\_2$ and $variable\_5 = 1$ determine the values of $variable\_6$.

2) Evaluation Methods: In the experiments, the MDLGP has learnt both of the FDN and BN. The rules extracted from them were compared with those discovered by Generic Genetic Programming (CGP) and the C4.5 [15]. By the nature of GP, rules learnt by the CGP are likely to have overlappings.

The data sets were split into two parts. 66% of them were used for the learning and the rest were used for the evaluation. The experimental results are evaluated in terms of the classification accuracy and the number of rules based on 10 independent runs.

B. Results for Data Set 1

The values of the maximum number of generations, the tournament competition size, the number of individuals, the extinction portion, the number of discretization levels, the crossover rate, the mutation rate, the insertion rate and the deletion rate are 1000, 7, 50, 0.7, 10, 0.3, 0.3, 0.3 and 0.1 respectively.

Table III shows the results for Data Set 1. The rules learnt by the FDN have got the most accurate result. Both of the CGP and the C4.5 failed to discover good rules. The C4.5 has stuck into a local optimum, it was unable to discover the best decision tree shown in figure 3 and thus got the worst classification result.

Table IV shows the first five rules learnt by the CGP in a run. As the rules were evaluated and evolved separately, they had no cooperation and thus have many overlappings. For instance, it is hard to interpret the effective meaning of the fifth rule. The rules have the lower classification accuracy and the lower understandability. Furthermore, as it is hard to realize their effective meaning, we could not anticipate which of them would be (or would not be) fired ultimately. So all of them were preserved and were collected to be the rule set. In result, the CGP had the largest fixed number of rules.

C. Results for Data Set 2

The values of the maximum number of generations, the tournament competition size, the number of individuals, the extinction portion, the number of discretization levels, the crossover rate, the mutation rate, the insertion rate and the deletion rate are 1000, 7, 50, 0.7, 10, 0.3, 0.3, 0.3 and 0.1 respectively.

Table V shows the results for Data Set 2. The rules discovered by the FDN have got the most accurate result. The BN and the C4.5 could not learn the rules classifying the data items accurately, since they could not represent the higher-order relationship existing in Data Set 2. In contrast, the FDN can represent any kind of relationships and got the most accurate result.

D. Results for Data Set 3

The values of the maximum number of generations, the tournament competition size, the number of individuals, the extinction portion, the number of discretization levels, the crossover rate, the mutation rate, the insertion rate and the deletion rate are 1000, 7, 50, 0.7, 10, 0.3, 0.3, 0.3 and 0.1 respectively.

Table VI shows the results for Data Set 3. Both of the FDN and the BN have got the best classification results, but the latter had the larger number of rules. Tables VII and VIII show the first rule learnt by the FDN and the first five ones discovered by the BN in a run respectively. Their rules have the same total meaning. Instead of enumerating all of the instances of the combination of the conditions, the FDN represented the higher-order relationships directly and thus produced the smallest number of rules. The smaller number of rules increases the rules' understandability and decreases the usage difficulty for expert systems. Similarly, the C4.5 could not represent higher-order relationships, so it generated a number of rules too.

The experimental results show the FDN, BN and decision trees guarantee to produce rules with no overlapping. But, as decision tree learning has much larger search space, algorithms may stuck into local optima and cannot discover the best solution. On the other hand, BN lacks of the ability to handle continuous, interval and ordinal values and cannot represent higher-order relationships among variables, so it is likely to have higher number of rules and cannot get the best classification result. While compared to decision trees, the FDN has the smaller learning search space; it can also handle all kinds of values; and it can represent higher-order relationships among variables. Therefore, the FDN and the MDLGP together is the best method to learn non-overlapping rules.

IV. Conclusion

In this paper, we propose to use the Functional Dependency Network and the MDL Genetic Programming to learn classification rules without overlappings. Rules with no overlapping have higher understandability and lower usage difficulty for expert systems. The experimental results show the proposed method has got the best results.

Acknowledgment

This research was partially supported by RGC Earmarked Grant 4192/03E of Hong Kong SAR and RGC Research Grant Direct Allocation of the Chinese University of Hong Kong.

References

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C.J. Stone, editors. *Classification and Regression Trees.* CRC Press, 1984.

[2] Alex A. Freitas, editor. *Data Mining and Knowledge Discovery with Evolutionary Algorithms.* Springer, 2002.

[3] Joao Gama. Discriminant trees. In *Proceeding 16th International Conf. on Machine Learning*, pages 134–142, 1999.

[4] G.W. Greewood, G.B. Fogel, and M. Ciobanu. Emphasizing extinction in evolutionary programming. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 666–671, 1999.

TABLE III

RESULTS FOR DATA SET 1.

	Classification Accuracy				Number of rules
	Average(%)	Best(%)	Worst(%)	Standard Deviation	Average
MDLGP with FDN	96.28	98.36	94.36	0.01	12
MDLGP with BN	95.79	97.81	93.99	0.01	12
CGP	74.64	77.05	72.13	0.02	50
C4.5	63.66	67.30	59.00	0.03	14

TABLE IV

THE FIRST FIVE RULES LEARNT BY THE GENERIC GENETIC PROGRAMMING FOR DATA SET 1.

1. if (var_5=var_1) and (var_1=var_4) then (class=1)
2. if (var_5$\neq$var_1) and (var_1$\neq$var_4) and (var_1$\neq$var_2) and (var_5$\neq$var_2) then (class=0)
3. if (var_2=var_1) and (var_4=var_5) then (class=1)
4. if (var_5=var_1) and (var_1=var_4) then (class=0)
5. if (var_5$\neq$var_1) and (var_1$\neq$var_4) and (var_4$\neq$var_2) and (var_1$\neq$var_2) and (var_3=var_5) and (var_0=var_3) then (class=0)

TABLE V

RESULTS FOR DATA SET 2.

	Classification Accuracy				Number of rules
	Average(%)	Best(%)	Worst(%)	Standard Deviation	Average
MDLGP with FDN	100.00	100.00	100.00	0.00	3
MDLGP with BN	61.30	65.70	56.52	0.03	5
CGP	98.36	100.00	92.27	0.03	50
C4.5	78.21	83.10	71.80	0.03	34

TABLE VI

RESULTS FOR DATA SET 3.

	Classification Accuracy				Number of rules
	Average(%)	Best(%)	Worst(%)	Standard Deviation	Average
MDLGP with FDN	100.00	100.00	100.00	0.00	8
MDLGP with BN	100.00	100.00	100.00	0.00	36
CGP	81.96	84.24	79.89	0.01	50
C4.5	96.59	100.00	91.10	0.04	28

TABLE VII

THE FIRST RULE LEARNT BY THE FUNCTIONAL DEPENDENCY NETWORK FOR DATA SET 3.

1. if (var_5=1) and (var_1=var_2), then (class=1)

TABLE VIII

THE FIRST THREE RULES LEARNT BY THE BAYESIAN NETWORK FOR DATA SET 3.

1. if (var_5=1) and (var_1=1) and (var_2=1), then (class=1)
2. if (var_5=1) and (var_1=2) and (var_2=2), then (class=1)
3. if (var_5=1) and (var_1=3) and (var_2=3), then (class=1)

[5] S. Hettich, C.L. Blake, and C.J. Merz. Uci repository of machine learning databases, 1998.

[6] JH Holland. Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In *Machine Learning*, pages 593–623, 1986.

[7] De Jong KA, Spears WM, and Gordon DF. Using genetic algorithms for concept learning. In *Machine Learning*, pages 161–188, 1993.

[8] J.R. Koza, editor. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

[9] RuleQuest Research Pty Ltd. Data mining tools see5 and c5.0. http://www.rulequest.com/see5-info.html, 2003.

[10] Z. Michalewic, editor. *Genetic Algorithms+DataStructures=Evolution Programs.* New York: Springer-Verlag, 1994.

[11] Durga Prasad Muni, Nikhil R. Pal, and Jyotirmoy Das. A novel approach to design clasifiers using genetic programming. In *IEEE Transactions on Evolutionary Computation*, pages 183–195, 2004.

[12] Sreerama K. Murthy, Simon Kasif, and Steven Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2:1–32, 1994.

[13] P. Myllymaki, T. Silander, H. Tirri, and P. Uronen. B-course: a web service for bayesian data analysis. In *Proceedings of the 13th International Conference on Tools with Artificial Intelligence*, pages 247–256, 2001.

[14] J.R. Quinlan, editor. *C4.5: Programs for Machine Learning*. Morgan Kauffman, 1993.

[15] Ngan P. S., Wong M. L., Lam W., Leung K. S., , and J. C. Y. Cheng.

Medical data mining using evolutionary computation. In *Artificial Intelligent in Medicine, Special Issue On Data Mining Techniques and Applications in Medicine*, pages 73–96, 1999.

[16] Wing-Ho Shum, Hui-Dong Jin, Kwong-Sak Leung, and Man-Leung Wong. A self-organizing map with expanding force for data clustering. In *ICDM02, IEEE International Conference on Data Mining*, pages 434–441, 2002.

[17] Wing-Ho Shum, Kwong-Sak Leung, and Man-Leung Wong. Learning functional dependency networks based on genetic programming. In *ICDM05, IEEE International Conference on Data Mining*, pages 232–230, 2005.

[18] R.S. Siegler. Three aspects of cognitive development. In *Cognitive Psychology*, pages 481–520, 1976.

[19] S. Sohn and C.H. Dagli. Advantages of using fuzzy class memberships in self-organizing map and support vector machines. In *Proceedings. IJCNN '01. International Joint Conference on Neural Networks*, pages 1886–1890, 2001.