Learning acyclic decision trees with 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

One objective of data mining is to discover parent-child relationships among a set of variables in the domain. Moreover, showing parents' importance can further help to improve decision makings' quality. Bayesian Network (BN) is a useful model for multi-class problems and can illustrate parent-child relationships with no cycle. But it cannot show parents' importance. In contrast, decision trees state parents' importance clearly, for instance, the most important parent is put in the first level. However, decision trees are proposed for single-class problems only, when they are applied to multi-class ones, they are likely to produce cycles representing tautologic. In this paper, we propose to use MDL Genetic Programming (MDLGP) and Functional Dependency Network (FDN) to learn a set of acyclic decision trees [9]. The FDN is an extension of BN; it can handle all of discrete, continuous, interval and ordinal values; it guarantees to produce decision trees with no cycle; its learning search space is smaller than decision trees'; and it can represent higher-order relationships among variables. The MDLGP is a robust Genetic Programming (GP) proposed to learn the FDN. We also propose a method to derive acyclic decision trees from the FDN. The experimental results demonstrate that the proposed method can successfully discover the target decision trees, which have no cycle and have the accurate classification results.

1 Introduction

Decision tree is a powerful classification model for single-class problems and can show parents' importance. Well-known decision tree learning algorithms include OC1, Ltree, C4.5 and C5.0 [7, 8, 1, 5]. Forests of decision trees were also proposed and were applied to different areas [11, 4]. However, all of them cannot manage multi-class problems and thus have no method to avoid cycles.

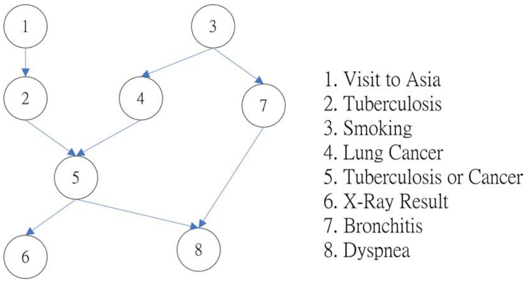


Figure 1. The Bayesian Network for Asia.

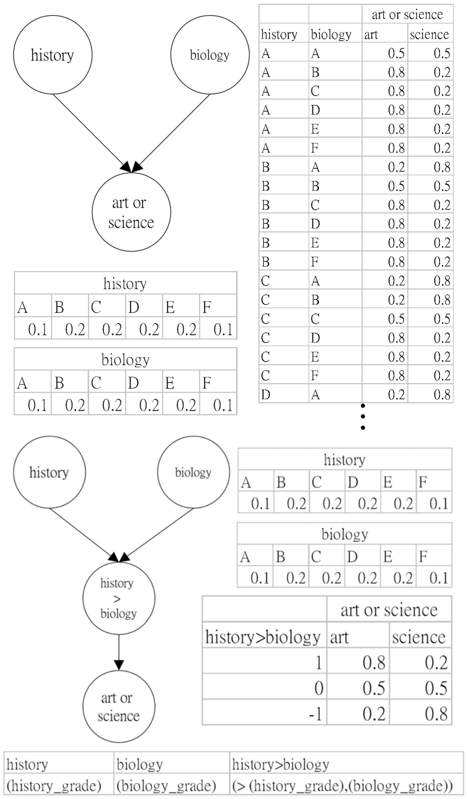
Figure 6 shows an example of decision trees with cycles. From the decision trees, it is unable to deduce $history$ and $biology$'s values, as they are determined by each other, i.e. several cycles exist and thus, the decision trees are useless for decision makings.

In contrast, BN were proposed for multi-class problems and represent acyclic parent-child relationships with, but cannot illustrate parents' importance.

Figure 1 shows the BN for the benchmark data set, Asia [3]. For instance, it illustrates the variable, $Dyspnea$ has two parents, $Tuberculosis\_or\_Cancer$ and $Bronchitis$, but cannot show which one is more important (or if they have the same importance). If medical doctors can know which one is more important, they would able to make better decisions. Therefore, showing parents' importance can help to improve decision makings' quality.

Furthermore, BN can handle discrete values only, continuous, interval and ordinal ones must be discretized and thus, the order information is lost. The disability to handle continuous, interval and ordinal values increases the network complexity; the relationships represented by BN become less understandable; and higher-order relationships among variables cannot be learnt.

Proceedings of the International Multi-Conference on Computing in the Global Information Technology (ICCGI'06)
0-7695-2629-2/06 $20.00 © 2006 IEEE



		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

⋮

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

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))

Figure 2. (Upper) The Bayesian Network for Programme Selection. (Lower) The Functional Dependency Network for Programme Selection.

We propose to use the MDLGP and the FDN to learn acyclic decision trees [9]. 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, it represents acyclic parent-child relationships and has smaller learning search space than decision trees' [10].

Consider an university programme selection problem that has two ordinal variables, 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 has done better in history; otherwise, he/she would study one randomly. Figure 2 (upper) shows the BN representing the problem. The largest conditional table on the right states $history$ and $biology$ are the parents of $art\_or\_science$. 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.

Figure 2 (lower) shows the FDN for the programme selection problem. The FDN has a functional node, $history > biology$. It represents the grade comparison between the subjects and its conditional table has only one entry specifying the 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 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 relationships; and the relationships can be expressed in rule and tree formats easily.

The MDLGP is a GP proposed to learn the FDN, which uses an extended MDL to evaluate 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 build acyclic decision trees from the FDN. The paper is organized as follows. We introduce the FDN in Section 2.1, followed by descriptions of the MDLGP and the decision tree building procedure. The experimental results are presented in Section 3. A conclusion is given in the last section.

2 The Algorithm

2.1 The Functional Dependency Network

The FDN is a directed acyclic network. A variable node denotes a variable in the domain and a directed link represents the dependency relationships between the child and the parents. Each variable node has a conditional table specifying the probability of each particular value of the variable node given an instantiation of the parents. For each variable node with no parent, the conditional table specifies the priori probability distribution.

The FDN 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 rep-

0-7695-2629-2/06 $20.00 © 2006 IEEE

resents an interval. To generate a continuous value, 1) selecting 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.

2.2 The MDL Genetic Programming

2.2.1 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 2 (lower) is $((> \ history \ biology) \rightarrow art\_or\_science)$

The MDLGP uses a grammar to prevent the closure problem [9]. It 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.

2.2.2 Evaluation, Selection and Reproduction

The MDLGP uses the extended MDL to evaluate individuals and the smaller MDL score is the better [9].

It 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 by Roulette Wheel method [6].

The MDLGP has four genetic operators, they are the mutation, the crossover, the insertion and the deletion [9]. 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 [2].

2.3 Decision Trees Building

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, decision trees are derived from it.

Firstly, for each of the variables having parents, build a decision tree table recursively. The decision tree tables are very similar to the conditional ones, except the entries inside are ordered according to the information gain calculations and entries in the same column are no longer necessary to represent the same parent. If a set T of data items is partitioned into disjoint exhaustive classes $c_1, c_2 ..., c_j$ on the basis of the value of a variable, then the information needed to identify the class of an element of T is,

$$info(T) = \sum_{i=1}^{n} \frac{T_i}{T} * info(T_i) \qquad (1)$$

where j is the number of classes and

$$info(T_i) = -\sum_{j} \frac{c_i}{T_i} * log(\frac{c_i}{T_i}) \qquad (2)$$

To build the decision tree tables, for each of the variables having parents, 1) starting with a empty decision tree table and setting the first column as the working one; 2) setting the whole data set as the working data partition and setting all the rows are the working ones; 3) calculating the parents' $info(T)$ and selecting the one with the highest value; 4) if the selected parent is not the only one left, partition the working data partition on the basis of the selected parent's values and then to correspond to the new data partitions, divide the working rows evenly. Otherwise goto the step 7; 5) for each of the new data partitions, put the selected parent and the value into the entries in the working column and the corresponding working rows; 6) for each of the new data partitions, set the next column and the corresponding rows as the working ones. Then, go back to the step 3 without the selected parent; 7) calculating the possibilities for the class's values.

Secondly, the decision trees are derived from the decision tree tables. For each of the decision tree tables, 1) starting with a empty decision tree, setting the first level as the working one; 2) setting the first column of the decision tree table as the working one; 3) for each set of consecutive entries containing the same parent and the same value in the working column, add a node to the working level; 3) if the new nodes are not in the first level, then connect them to theirs parents, which are the nodes represented by the entries in the same rows but the previous column; 4) if it is not the last column, then set the next column and the next level as the working ones and go back to the step 3; 5) for each of the rows, among the class's values, select the one with the highest possibility as the leaf; 6) if there are more than one class values having the highest probability, select one randomly; and 7) finally, neighbor branches that having the same parent, the same variables and the same class's values are combined, to produce a concise and more understandable decision tree.

Figure 3 shows an example of the decision tree building. The uppermost table shows the characteristics of a synthetic data set, which has 2 variables, 400 data items and 1 class. Both of $variable\_A$ and $variable\_B$ are the parents of $class$. For instance, the table shows there are 101 data items with $variable\_A$ is 1 and $class$ is 0. Firstly, a decision tree table is built recursively. According to the information gain calculations, $variable\_B$ is the most important parent and thus, it and its values are put into the first column. Next, for each of the values, we determine the second most important parent. Since $variable\_A$ is the only one left, it and its values are put into the corresponding entries in the second column. Thirdly, the decision tree is derived from the decision tree table and the parent in the first column becomes the node in the first level. Then, for each set of consecutive entries containing the same parent and the same value in the column, child nodes are added, which are those represented by the entries in the same rows but the next column. When there is no more parent left, the leaves, i.e. $class$ and its values are added. For each of the branches, the $class$'s value with the highest probability is selected. If there are more than one $class$'s values having the highest probability, one is selected randomly. For example, when $variable\_A$ is 0 and $variable\_B$ is 1, the probabilities to have $class$ is 0 and is 1 are the same, i.e. 0.5, so either one, i.e. 0 is chosen. Finally, neighbor branches having the same parent, the same variables and the same class's values are combined. For instance, in the branch of $variable\_B$ is 0, both of the $class$'s values are 0 when $variable\_A$ is 0 and is 1, so they are merged.

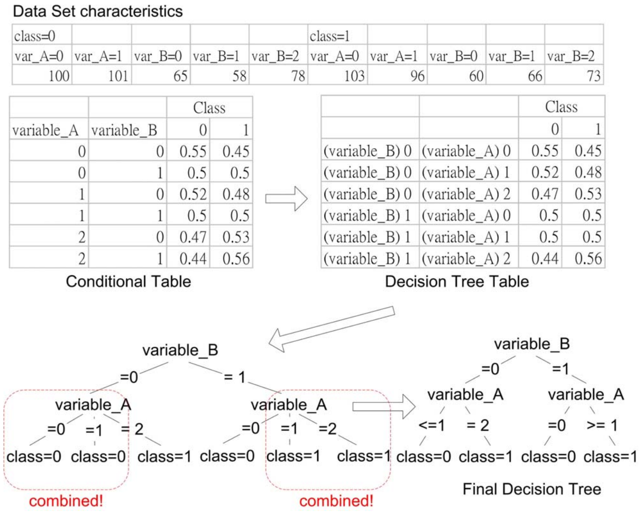
3 Experimental results

3.1 Experimental settings

3.1.1 Data Sets

We evaluated the proposed method on two data sets. The first one is Monk 1 from UCI machine learning repository [3]. It has 7 nominal variables, 556 data items and $variable\_6$ is the class.

The other one is Programm Selection used in [9]. It consists of 7 continuous, ordinal and discrete variables. It has 1000 data items and $history$, $biology$ and $art\_or\_science$ are the classes.



Data Set characteristics

class=0					class=1				
var_A=0	var_A=1	var_B=0	var_B=1	var_B=2	var_A=0	var_A=1	var_B=0	var_B=1	var_B=2
100	101	65	58	78	103	96	60	66	73

		Class	
variable_A	variable_B	0	1
0	0	0.55	0.45
0	1	0.5	0.5
1	0	0.52	0.48
1	1	0.5	0.5
2	0	0.47	0.53
2	1	0.44	0.56

Conditional Table

		Class	
		0	1
(variable_B) 0	(variable_A) 0	0.55	0.45
(variable_B) 0	(variable_A) 1	0.52	0.48
(variable_B) 0	(variable_A) 2	0.47	0.53
(variable_B) 1	(variable_A) 0	0.5	0.5
(variable_B) 1	(variable_A) 1	0.5	0.5
(variable_B) 1	(variable_A) 2	0.44	0.56

Decision Tree Table

Figure 3. An example of the decision tree building.

3.1.2 Evaluation Methods

In the experiments, the MDLGP learnt both of the FDN and BN. Then, decision trees are derived from the discovered networks. Lastly, the discovered decision trees were compared with those learnt by the Ltree and the C4.5.

The Ltree and the C4.5 can only learn a decision tree for one class each time, so we executed them ten times, for each of the classes.

Furthermore, the FDN and BN can determine which variables are the classes, but we have to specify them for the Ltree and the C4.5.

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 decision trees were evaluated if they have cycles and were also compared in terms of the classification accuracy based on 10 independent runs.

3.2 Results for Monk 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 1 shows the results for Monk 1. The FDN has got the highest classification accuracy. As the BN, the Ltree and the C4.5 could not learn the higher-order relationship in the data set, they got the lower accuracies and the more complicated decision trees. Moreover, since decision tree

Table 1. Results for Monk 1.

	Average(%)	Best(%)	Worst(%)	Standard Deviation
MDLGP with FDN	96.28	98.36	94.36	0.01
MDLGP with BN	95.79	97.81	93.99	0.01
Ltree	91.91	100.00	81.97	0.07
C4.5	93.92	100.00	83.60	0.05

learning has larger search space, the Ltree and the C4.5 have stuck into the local optimum and got the worst results.

3.3 Results for Programme Selection

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 2 shows the results for Programm Selection. The FDN has got the best result for $art\_or\_science$ and the second highest accuracies for $history$ and $biology$. Since both of the BN and the C4.5 could not represent the higher-order relationships in the data set, they got the worse results. Besides, although the Ltree had the best results for $history$ and $biology$, it produced cyclic decision trees which are useless for decision makings.

Figures 4, 5, 6 and 7 show the decision trees learnt by the FDN, the BN, the Ltree and the C4.5 respectively. Since the decision trees are very large, only parts of them are shown. The FDN and the BN have learnt the acyclic decision trees successfully. As the FDN can represent higher-order relationships, it does not need to enumerate the instances of the combination of the parents and thus produced the smallest decision trees. In contrast, both of the Ltree and the C4.5 were failed to learn decision trees with no cycle. For example, the Ltree's results show the values of $history$ and $biology$ are determined by each other!

4 Conclusion

In this paper, we propose to use the MDLGP and the FDN to learn acyclic decision trees for multi-class problems. BN can handle multi-class problems, but cannot handle higher-order relationships and cannot illustrate parents' importance. In contrast, decision trees can show parents' importance, but can manage single-class problems only. The FDN can handle all kinds of values; it guarantees to produce decision trees with no cycle; it has smaller learning search space; and it can represent higher-order relationships among variables. The experiments demonstrated that the proposed method can successfully learn the target acyclic decision trees.

References

[1] J. Gama and P. Brazdil. Linear tree. In *Intelligent Data Analysis*, pages 1–22, 1999.

[2] 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.

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

[4] C.Z. Janikow. Fuzzy decision forest. In *22nd International Conference of the North American Fuzzy Information Processing Society, 2003.*, pages 480–483, 2003.

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

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

[7] 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.

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

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

[10] Wing-Ho Shum, Kwong-Sak Leung, and Man-Leung Wong. Learning non-overlapping rules a method based on functional dependency network and mdl genetic programming. In *CEC05, Proceedings of IEEE Congress on Evolutionary Computation*, 2006.

[11] Huimin Zhao and A.P. Sinha. An efficient algorithm for generating generalized decision forests. In *IEEE Transactions on Systems, Man and Cybernetics, Part A*, pages 754–762, 2005.

Table 2. Results for Programm Selection.

		Average(%)	Best(%)	Worst(%)	Standard Deviation
history	MDLGP with FDN	92.24	93.84	91.26	0.01
	MDLGP with BN	76.58	78.79	73.94	0.02
	Ltree	95.39	97.61	92.69	0.01
	C4.5	84.31	85.80	81.90	0.01
biology	MDLGP with FDN	91.15	93.94	88.18	0.02
	MDLGP with BN	77.55	80.61	73.33	0.02
	Ltree	96.10	97.46	94.63	0.01
	C4.5	84.68	87.80	79.40	0.02
art or science	MDLGP with FDN	71.13	72.42	71.36	0.01
	MDLGP with BN	64.76	69.70	61.21	0.02
	Ltree	71.00	72.54	68.06	0.01
	C4.5	68.23	70.30	65.80	0.01

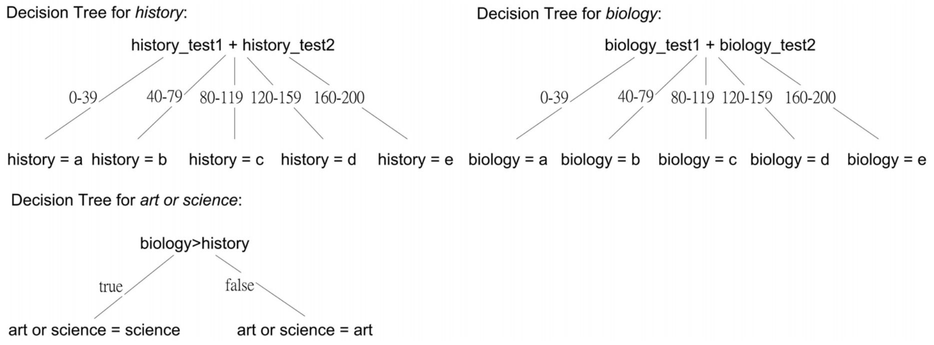


Figure 4. The decision trees learnt by the Functional Dependency Network for Programme Selection.

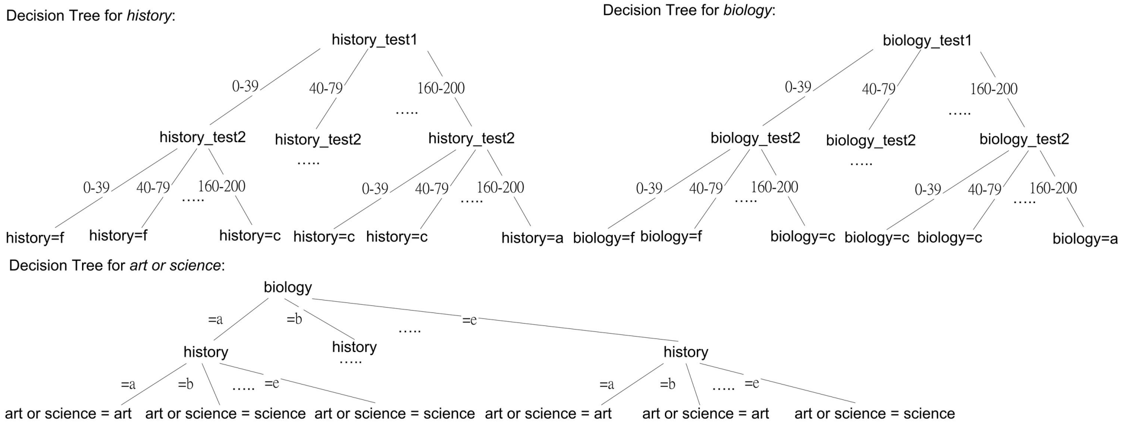
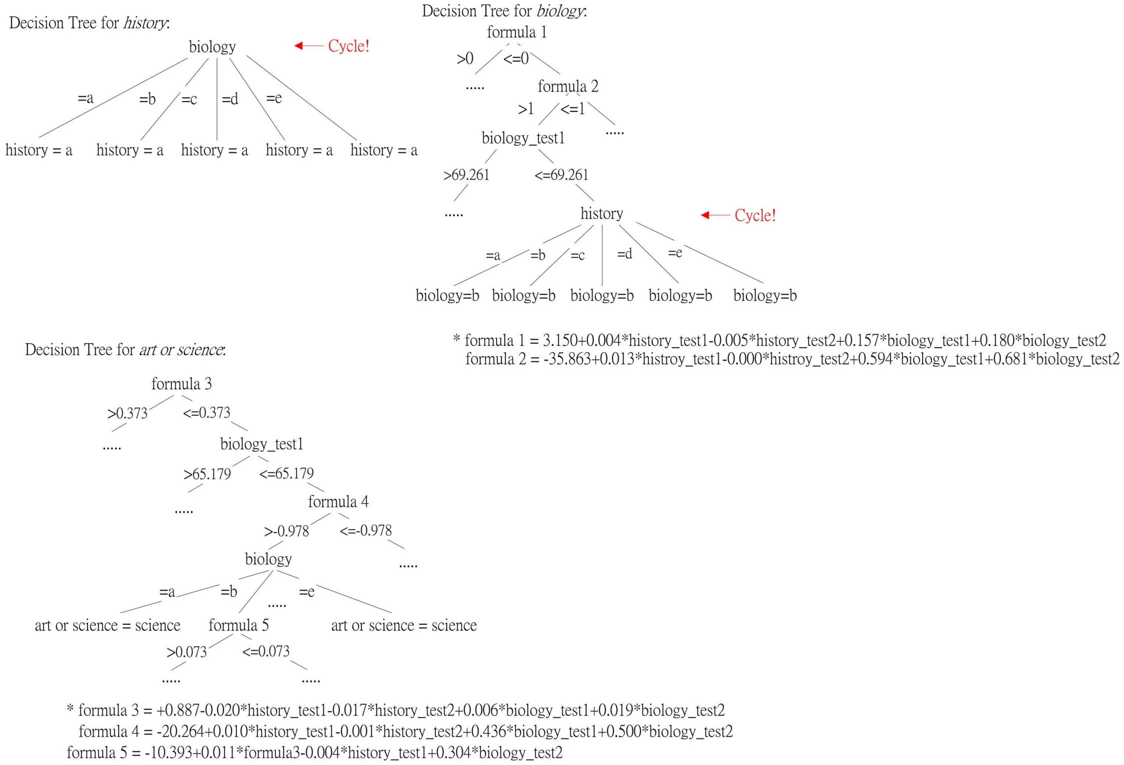


Figure 5. The decision trees learnt by the Bayesian Network for Programme Selection.



* formula 1 = 3.150+0.004*history_test1-0.005*history_test2+0.157*biology_test1+0.180*biology_test2
formula 2 = -35.863+0.013*histroy_test1-0.000*histroy_test2+0.594*biology_test1+0.681*biology_test2

* formula 3 = +0.887-0.020*history_test1-0.017*history_test2+0.006*biology_test1+0.019*biology_test2
formula 4 = -20.264+0.010*history_test1-0.001*history_test2+0.436*biology_test1+0.500*biology_test2
formula 5 = -10.393+0.011*formula3-0.004*history_test1+0.304*biology_test2

Figure 6. The decision trees learnt by the Ltree for Programme Selection.

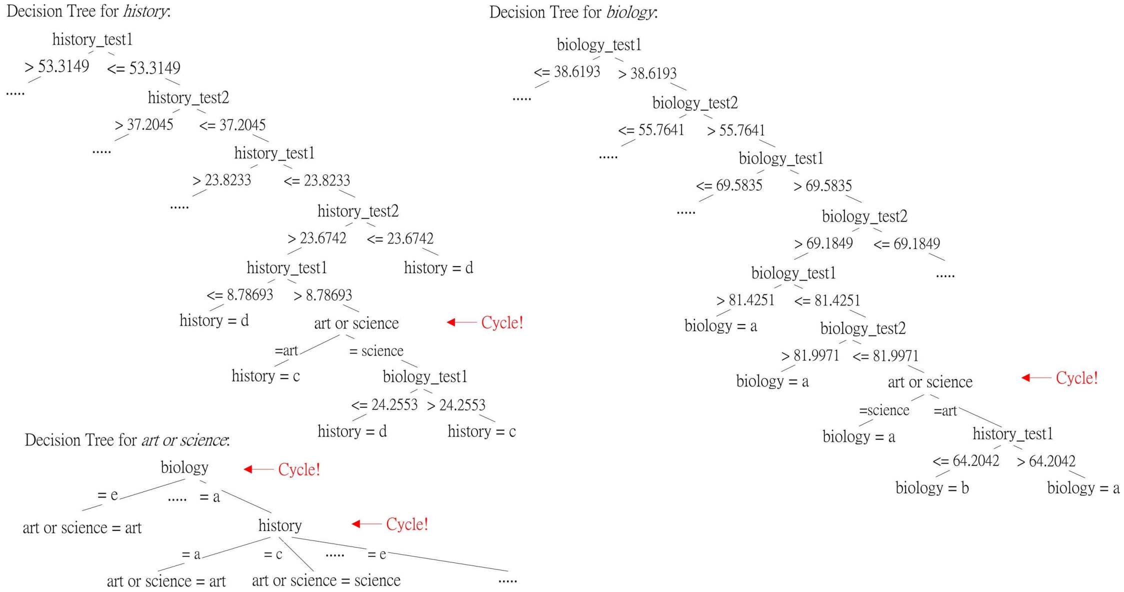


Figure 7. The decision trees learnt by the C4.5 for Programme Selection.