An Evolutionary Algorithm for Making Decision Graphs for Classification Problems

Shingo Mabu
Graduate School of Science and Engineering, Yamaguchi University, Tokiwadai2-16-1 Ube, Yamaguchi 755-8611, Japan

Masanao Obayashi
Graduate School of Science and Engineering, Yamaguchi University, Tokiwadai2-16-1 Ube, Yamaguchi 755-8611, Japan

Takashi Kuremoto
Graduate School of Science and Engineering, Yamaguchi University, Tokiwadai2-16-1 Ube, Yamaguchi 755-8611, Japan

E-mail: mabu@yamaguchi-u.ac.jp, m.obayas@yamaguchi-u.ac.jp, wu@yamaguchi-u.ac.jp

Abstract

As the exponential increase of data in the world, machine learning, pattern recognition, data mining etc. are attracting more attentions recently. Classification is one of the major research in pattern recognition and a large number of methods have been proposed such as decision trees, neural networks (NNs), support vector machines (SVMs). In order to easily understand and analyze the reason of the classification results, decision trees are useful comparing to NNs and SVMs. In this paper, to enhance the classification ability of decision trees, a new evolutionary algorithm for creating decision graphs is proposed as a superset of decision trees, where multi-root nodes and majority voting mechanism based on Maximum a posteriori are introduced. In the performance evaluation, it is clarified that the proposed method shows better classification ability than decision trees.

Keywords: evolutionary computation, decision graph, classification, majority vote, multi root nodes

1. Introduction

In the research fields of pattern recognition and machine learning, a large number of classification algorithms have been proposed[1], such as neural networks (NNs), support vector machines (SVMs)[1] and decision trees[2]. NNs and SVMs mathematically create decision boundaries to separate classes with high accuracy and have applied to many applications such as intrusion detection[3], prediction of stock market indexes[4]. As for the decision trees, typical learning algorithms are classification and regression tree (CART)[5], ID3 and C4.5[2]. NNs, SVMs and their extended algorithms can create distinguished decision boundaries for accurate classification, however, they are black box models, thus the reason of the classification results is difficult to be shown to the end-users. On the other hand, decision trees can visually show the rules of classification explicitly by the tree structures, therefore, it is useful when users want to know the created rules and analyze the relationships between attributes in databases. However, when dealing with databases with a large

number of attributes, many rules have to be created by one tree, then the size of the tree would become too large. A large tree structure would cause over-fitting to the training data and decrease the generalization ability, therefore, a pruning of tree is to be executed. To enhance the generalization abilities of decision trees, some extended algorithms have been also proposed[6].

In this paper, to enhance the representation ability of decision tree and obtain better classification accuracy, directed-graph based classifier with recurrent connections and its evolutionary algorithm are proposed. In the case of tree structure, the classification procedure starts from the root node, selects appropriate branches at non-terminal nodes, and finally classifies the data at the reached terminal node. Therefore, the node transition flows from the top layer to the bottom layers. In the case of graph structure, 1) any non-terminal nodes can become root nodes, and 2) the same nodes can be shared by several rules, therefore, many rules can be created by the smaller number of nodes and the program structure becomes quite compact.

In addition, this paper proposes some specific mechanisms to enhance the classification abilities of decision graph. First, multi root nodes are introduced to create various kinds of rules, while the standard decision tree uses one root node. Second, majority voting mechanism is introduced to enhance the generalization ability. In the proposed method, the classification process starts from one of the root nodes, selects branches at non-terminal nodes, and classifies a data into a certain class (vote for a certain class) at the transient terminal node. However, the transient terminal node has a connection to the next non-terminal node, therefore, the classification process continues and the same process is repeated until the fixed number of votes are executed. This multi-voting mechanism is expected to show better generalization ability.

The rest of this paper is organized as follows. In section 2, the basic structure of evolutionary decision graph and its gene expression is explained. In section 3, an evolutionary algorithm of decision graphs and how to classify data are explained. In section 4, after benchmark problems of classification and simulation conditions are explained, the results and analysis are described. Section 5 is devoted to conclusions.

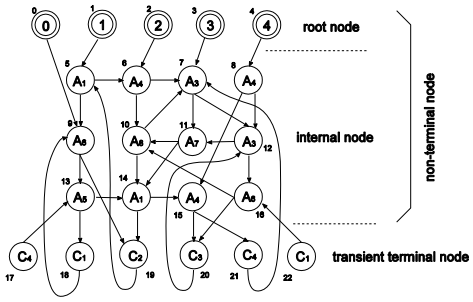

Fig. 1. Basic structure of evolutionary decision graph

2. Structure of Evolutionary Decision Graph

2.1. *Basic structure*

The basic structure of the decision graph is shown in Fig. 1, where there are three components: root nodes, internal nodes and transient terminal nodes. Here, root nodes and internal nodes are called non-terminal nodes. Root nodes have a function to start node transition and decide the next non-terminal node to which the current node transfers. The proposed decision graph has multi root nodes, and the node transition starts from each root node one by one, which results in considering various kinds of rules to make classification. The function of non-terminal nodes is the same as standard decision tree with multiple branches. In Fig 1, each non-terminal node has its own attribute A_i, $i \in \{1,2,\cdots,n\}$ to be judged, and when a data is inputted to the decision graph, an appropriate branch is selected according to the value of the data. The function of transient terminal node is to determine the class of the inputted data, and cast one vote to the class. For example, if the transient terminal node determines the class as "class 1", class 1 gets one vote. After the voting, the node transition continues depending on the connection from the transient terminal node. The detailed mechanism of making classification at each transient terminal node is explained in section 3.2.

2.2. *Gene Structure*

Fig. 2 shows the gene structure of node number k in a decision graph. NF_k shows a node function number: NF_k=0 is a root node, 1 is an internal node, and 2 is a transient terminal node. ATT_k shows an attribute number

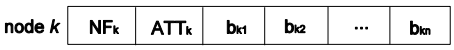

Fig. 2. Gene Structure of evolutionary decision graph

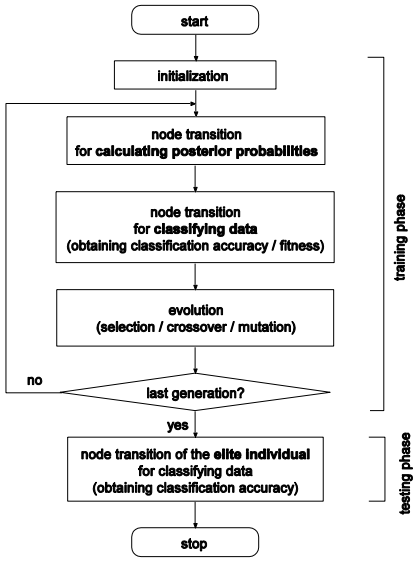

Fig. 3. Flowchart of training and testing phases in evolutionary decision graph

to be judged at non-terminal node k, or a class attribute to be assigned at transient terminal node k. b_{k1}, b_{k2}, ..., b_{kn} shows the next node number connected from node k. Transient terminal nodes have one branch to the next node, thus only b_{k1} is used at each node. Non-terminal nodes use b_{k1}, b_{k2}, ..., b_{kn} depending on the possible number of branches for ATT_k.

3. Evolutionary algorithm of decision graphs

Fig. 3 shows the flow of the whole implementation of evolutionary decision graph. In the training phase, after initializing the individuals, voting classes at transient terminal nodes are determined by posterior probabilities, and the classification accuracy for training data is obtained as the fitness for evolution. Then, evolutionary operations such as selection, crossover and mutation are carried out. The elite individual obtained in the final generation is picked up for the testing phase.

3.1. *Initialization of an individual*

First, the numbers of root nodes, internal nodes and transient terminal nodes are determined (NF_k is determined) depending on the complexity of the target problem. Second, the values of ATT_k are randomly determined. ATT_k of non-terminal nodes are determined by randomly selecting one of the attributes in a database. ATT_k of transient terminal nodes are determined by maximum posterior probabilities explained in section 3.2. Finally, the branch connections are determined as follows. b_{k1} of transient terminal nodes are determined by randomly selecting one node number from non-terminal nodes. b_{k1}, b_{k2}, ..., b_{kn} of non-terminal nodes are determined by randomly selecting one node number from all the nodes.

3.2. *Node transition for determining voting class at transient terminal nodes*

To classify a data into an appropriate class, which class is assigned to each transient terminal node is very important. In this subsection, the procedure of assigning voting classes to each transient terminal node based on maximum posterior probabilities is explained in detail.

First, how to execute node transition is explained before introducing the class assignment mechanism. When a training data (tuple) d in a database is inputted, the node transition starts from root node 0 (node number k=0). The attribute ATT_0 of data d is examined and one of the branches among b_{01}, b_{02}, ..., b_{0n} is selected (here, suppose b_{01} is selected). If the selected node b_{01} is a non-terminal node, the same procedure as the root node is executed to determine the next node. Fig. 4 shows an example of the branch selection, where, in Fig. 4 (a), the value of attribute ATT_k of data d is in the range of [0,0.3], and its corresponding branch is b_{k1}, thus the

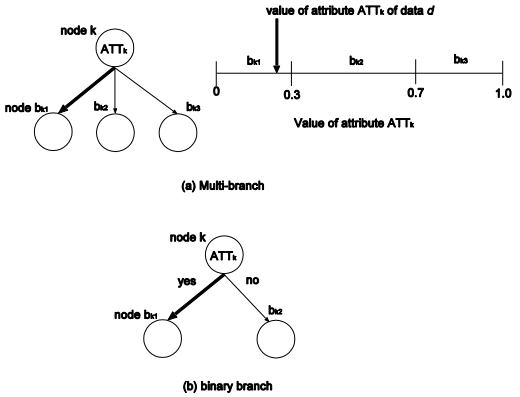

Fig. 4. An example of attribute division and branch decision

current node transfers to node b_{k1}. Fig. 4 (b) shows the case of binary branch where each non-terminal node executes yes-no branch selection. After repeating the branch selections at non-terminal nodes, if the current node reaches a transient terminal node t (Here, t shows the node number of transient terminal nodes), counter $N_i(t)$ increases by one, where i shows the class number of training data d. The current node transfers to the next node b_{k1} and continues the node transition until transient terminal nodes are visited predefined number of times, e.g., 10. However, if the node transition visits the same transient terminal nodes that have been already visited, the node transition restarts from the next root node, i.e., root node 1, to avoid the transition loop. In summary, the node transition starts from root node 0, and every time when a transition loop occurs, the node transition restarts from root node 1, 2, 3,... in the numerical order. By repeating the above node transition from the first data to the last one, the values of counters $N_i(t)$ of all the transient terminal nodes can be obtained. Then, the posterior probabilities of class i at transient terminal nodes t are calculated by

$$\mathrm{p}(\mathrm{C_i}|\mathrm{t}) = \frac{\mathrm{N_i(t)}}{\mathrm{N(t)}} \tag{1}$$

where, $N(t)$ is the number of data visiting node t.
Finally, voting class (ATT_t in Fig. 2) at transient terminal node t is determined by

$$\mathrm{ATT_t} = \arg\max_{\mathrm{j}} P\left(C_j|t\right) \tag{2}$$

3.3. *Node transition for classifying data*

After the procedure described in sections 3.1 and 3.2, the classification of data d is carried out and fitness for evolution is obtained as follows.

The procedure of node transition is the same as section 3.2 except that, at transient terminal nodes, voting to one of the classes is carried out instead of counting the number of visits $N_i(t)$. When node transition reaches a transient terminal node t, it casts a vote to class ATT_t, then the current node transfers to the next node b_{t1}. The node transition continues until the predefined number of voting finishes. Finally, the class that wins the highest votes becomes the classification result. The above process is repeated for all the data d, then the classification accuracy is calculated as the fitness of the individual.

3.4. *Genetic Operations in Evolutionary Decision Graphs*

The individuals are evolved by selection, crossover and mutation. The genetic operations of evolutionary decision graphs are based on Ref. 7. In crossover, two parent individuals are selected by tournament selection and randomly selected nodes with the probability of P_C are exchanged between the parents. In mutation, one individual is selected by tournament selection, and the connections, attributes in non-terminal nodes, and root nodes are randomly changed with the probability of P_m.

Table 1. Dataset information

dataset	# of data	# of attributes	# of classes
Heart disease	303	13	2
Sonar	208	60	2
Hepatitis	155	19	2
Wine	178	13	3
Breast cancer	569	30	2

Table 2. Simulation condition

# of generations	2000
# of individuals	503
crossover rate	0.1
mutation rate	0.02
# of nodes	200
root nodes	50
internal nodes	50
transient terminal nodes	100

4. Simulations

The classification accuracy is evaluated using benchmark datasets downloaded from UCI machine learning repository[8]. The information of the datasets is shown in Table 1. The objective of this paper is to show the advantage over the conventional decision tree, but for more reference, the comparison with support vector machine (SVM) is also implemented. The parameters of the proposed method are set as shown in Table 2. The

simulations are implemented by 10-fold cross validation, and the mean classification accuracy is calculated. In the training, some variations are given to the training dataset to avoid over fitting. In detail, 10% of the training data consists of the data misclassified by the elite individual in the previous generation, and the rest of 90% of the training data are generated by bootstrap sampling[9]. The continuous attributes in the original datasets are binarized by the Fayyad and Irani's entropy based method[10] executed by WEKA software[11] .

4.1. *Simulation results*

The performance evaluation is executed using the elite individual obtained in the last generation in the training phase. Table 3 shows the classification accuracy obtained by the proposed decision graph, decision tree and SVM. From Table 3, we can see that the mean accuracy obtained by decision graph is better than decision tree. In addition, decision graph obtains higher accuracy than decision tree in four datasets out of five. Comparing the accuracy obtained by decision graph with that by SVM, decision graph shows comparable result, and decision graph obtains higher accuracy than SVM in three datasets out of five. Although the mean accuracy for the five datasets is almost the same as SVM, decision graph can show the classification rules to users like decision tree, therefore, when we want to know and analyze the obtained rules, decision graph can be an useful method. In the future, we will execute simulations using more datasets and analyze the generated classification rules and characteristics of the decision graph.

Table 3. Testing results for benchmark datasets

	Classification accuracy [%]		
dataset	Decision graph	Decision tree	SVM
Heart disease	84.8	78.5	83.8
Sonar	82.2	71.2	76.0
Hepatitis	66.9	67.1	66.5
Wine	94.4	93.8	98.3
Breast cancer	94.9	93.1	97.7
mean	84.6	80.7	84.4

5. Conclusions

This paper proposed an evolutionary algorithm for creating decision graphs for classification problems. To enhance the classification abilities of the decision graph, some specific mechanisms such as multi-root nodes, unique genetic operators and majority voting were devised. From the simulation results, it was clarified that the proposed evolutionary algorithm improve the fitness (accuracy) of the individuals, and showed better classification accuracy comparing with decision tree and comparable accuracy to SVM. In the future, we will consider to apply impurity measures to the evaluation function of the evolutionary algorithm to clearly determine the decision boundaries of classes and improve the classification accuracy.

References

1. C. M. Bishop, et al., *Pattern recognition and machine learning* (Springer New York, 2006).
2. J. R. Quinlan, *C4.5: programs for machine learning*, (Morgan kaufmann, 1993).
3. S. J. Horng, M. Y. Su, Y. H. Chen, T. W. Kao, R. J. Chen, J. L. Lai, and C. D. Perkasa, A novel intrusion detection system based on hierarchical clustering and support vector machines, *Expert systems with Applications* **38**(1) (2011) 306–313.
4. E. Guresen, G. Kayakutlu, and T. U. Daim, Using artificial neural network models in stock market index prediction, *Expert Systems with Applications*, **38**(8) (2011) 10389–10397.
5. L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, Classification and regression trees, (CRC press, 1984).
6. L. Breiman, Random forests, *Machine learning*, **45(**1) (2001) 5–32.
7. S. Mabu, K. Hirasawa, and J. Hu, A graph-based evolutionary algorithm: Genetic network programming (GNP) and its extension using reinforcement learning, *Evolutionary Computation*, **15(**3) (2007) 369–398.
8. UCI machine learning repository. Online available: archive.ics.uci.edu/ml/
9. Bradley Efron and Robert. J. Tibshirani, An Introduction to the Bootstrap, (Springer, 1993).
10. U. M. Fayyad and K. B. Irani, Multi-interval discretization of continuous valued attributes for classification learning, in *Proc. of the 13th International Joint Conference on Artificial Intelligence*, (Morgan Kaufmann, 1993), pp. 1022–1027.
11. Waikato environment for knowledge analysis, open source project for machine learning, Online available: www.cs.waikato.ac.nz/ml/weka/