Journal of the Operations Research
Society of Japan
2006, Vol. 49, No. 1, 66-82

EXPLANATORY RULE EXTRACTION BASED ON THE TRAINED NEURAL NETWORK AND THE GENETIC PROGRAMMING

Jianjun Lu
Kyushu University

Shozo Tokinaga
Kyushu University

Yoshikazu Ikeda
Shinshu University

(Received January 17, 2005; Revised August 15, 2005)

Abstract This paper deals with the use of neural network rule extraction techniques based on the Genetic Programming (GP) to build intelligent and explanatory evaluation systems. Recent development in algorithms that extract rules from trained neural networks enable us to generate classification rules in spite of their intrinsically black-box nature. However, in the original decompositional method looking at the internal structure of the networks, the comprehensive methods combining the output to the inputs using parameters are complicated. Then, in our paper, we utilized the GP to automatize the rule extraction process in the trained neural networks where the statements changed into a binary classification. Even though the production (classification) rule generation based on the GP alone are applicable straightforward to the underlying problems for decision making, but in the original GP method production rules include many statements described by arithmetic expressions as well as basic logical expressions, and it makes the rule generation process very complicated. Therefore, we utilize the neural network and binary classification to obtain simple and relevant classification rules in real applications by avoiding straightforward applications of the GP procedure to the arithmetic expressions. At first, the pruning process of weight among neurons is applied to obtain simple but substantial binary expressions which are used as statements is classification rules. Then, the GP is applied to generate ultimate rules. As applications, we generate rules to prediction of bankruptcy and creditworthiness for binary classifications, and the apply the method to multi-level classification of corporate bonds (rating) by using the financial indicators.

Keywords: Algorithm, rule extraction, neural networks, genetic programming, prediction of bankruptcy, bond rating

1. Introduction

Numerous methods have been proposed in the literature to develop decision making and decision tables models. These models include traditional statistical methods such as multivariate discriminant function, logistic regression and even neural network techniques [5][21]. There are also methods categorized to classification tree such as the entropy-based decision tree and statistical procedure to generate linguistic if-then rules (inductive learning) [3].

However, most of these studies focus primarily on developing classification models with high perspective accuracy without any attention to explaining how the classifications are being made [3]. The explanation rather than classification accuracy plays a pivotal role in many fields such as financial applications where the evaluator may be required to give a justification for a certain credit application that is approved or rejected.

This paper deals with the use of neural network rule extraction techniques based on the Genetic Programming (GP) to build intelligent and explanatory evaluation systems [1][13] [20]. Though neural networks have their universal approximation property that seems to be attractive at first sight, their intrinsically black-box nature prevent them being successfully applied in various field. Fortunately, recent development in algorithms that extract rules

from trained neural networks enable us to generate classification rules [4][8][19]. However, the comprehensive mathematical method which relate the output to the inputs using the set of weight, bias and nonlinear activation functions are hard for human to trace. Then, in our paper, we utilized the GP to automatize the rule extraction process in the trained neural networks where the decision basically boils down to a binary classification problems.

In previous papers, we have successfully applied the GP to approximate the chaotic dynamics, where the approximated expression of dynamics is also utilized to control chaotic behavior [9]-[11][24]. We have also applied the GP to emulate the agents' behavior in artificial markets where agents use learning based on the production rules generated by the GP [6][7][12].

Among recent developments in algorithms that extract rules from trained neural networks, popular techniques such as Neurorule, Trepan and Nefclass are known [4][8][19]. Even though Neurorule is looking at the internal structure of networks, the process for extracting the rules is complicated. Techniques using fuzzy inference and fuzzy sets prevent us to represent rules in ordinary usable forms like decision tables. Then, we focus on the capability of trained neural networks for the discretization and decomposition of input variables to reduce simplified rules. The processes to combine the input and output units are carried out by using the GP. After training neural networks using the discretized input variables, the pruning process of weight among neurons is applied to obtain simple but substantial binary expressions which are used as statements is classification rules. Then, the GP is applied to generate ultimate rules. As applications, we generate rules to predict bankruptcy, to assess personal loans, and classify corporate bonds (rating) by using the financial indicators. The result shows that the rule generation proposed in the paper provide us a comparable performance for the decision making as the conventional results, and presents linguistically understandable expressions for human experts.

In the followings, in Section 2, we describe the relation between the production rule and the GP. Section 3 shows the algorithm for applying the GP to rule generations. In Section 4, we show the application of the method of the paper to the rule induction for bankruptcy prediction, personal loans, and bond rating.

2. Neural Networks and Rule Extraction

2.1. Rule extraction methods based on neural networks

Among recent developments in algorithms that extract rules from trained neural networks, three popular techniques are contrasted, namely, Neurorule, Trepan and Nefclass [4][8][19].

Nefclass has the architecture of a three-layered fuzzy perecptron, whereby the weights now represent fuzzy sets and the activation functions are now fuzzy set operators, different from classical multilayer perceptron [19]. The technique is often referred to as neurofuzzy systems [23]. Obviously, the above procedure will result in a large number of hidden neurons and fuzzy rules. Then, we need to specify maximum number of hidden neurons and best k rules before tuning neural networks to improve the classification accuracy. Even though Nefclass enforces all connections in representing the same linguistic label associated with the fuzzy sets, but it is hard to represents rules in ordinary usable forms like decision tables.

Trepan uses a pedagogical algorithm to extract decision trees from trained neural networks based on conventional symbolic learning [8]. At each step, a queue of leaves is further expanded into subtrees until stopping criterion is met. Different form conventional decision tree reduction algorithms where the number of available training observation becomes fewer and fewer along the split of tree, relabelled trained dataset is used to initiate the tree-

growing process. Furthermore, Trepan also enrich the training data with additional training instances relabelled by the neural networks. Then, Trepan allows splits with at least M-of N type of tests to generating (C_1 and C_2) or (C_1 and C_3) type rules. However, these M-of-N splits are constructed by using the heuristic search procedure. In the search procedure, we look at appropriate addition of new conditions and new threshold for variables.Evenmore, Trepan tries to simplify the rules by investigating whether M can be reduced without significantly degrading the information gain. These heuristic and twofold approach lead us to complicated algorithm.

Neurorule utilizes the decomposition approach by extracting rules at the level of the individual hidden and output units by analyzing the activation values, weight and biases [4]. At first, the input data represented in continuous numerical data and nominal data are discretized by adjusting threshold values for discretization. Then neural networks are trained to improve the classification accuracy by using the discretized input. Since the number of connections among neurons are large, redundant connections are removed while maintaining the accuracy of the networks. After pruning the redundant connections among units, the level of hidden layers is also discretized. Then, the relation between the input variables which are already discretized and the activation levels in hidden layers are formalized as rules. In a similar manner, the relation between the activation levels and the output variables are written as rules. Finally, these two sets of rules are summarized as a comprehensive set of rules. Indeed, Neurorule is looking at the internal structure of networks, the process for extracting the rules is complicated, and indirect with respect to the discretizartion and logical propositions. Even more, the obtainable rules are restricted to forms combined with logical operators "AND", and no extension for more general cases is expected.

2.2. Algorithm of neural networks

Neural networks are mathematical representations inspired by the functioning of the human brain [5][21]. Especially, the multilayer neural network is typically composed of an input layer, one or more hidden layers, and an output layer, each consisting of several neurons. Each neuron processes its input and generates one output value which is transmitted to the neurons in the subsequent layer. All neurons and layers are arranged in a feedforward manner, and no feedback connections are allowed. The output of the neuron i is computed by processing the weighted inputs and its bias term as follows.

In the formula, the $w_{i,j}^{n,n-1}$ denotes the weight connecting the j th unit in layer n with the i th unit in layer $n-1$. The value z_j^{n-1} is the output of jth unit in layer $n-1$. Then, the input to i th neuron in layer n is obtained by

$$u_i^n = \sum w_{i,j}^{n,n-1} z_j^{n-1} \tag{1}$$

The output of neurons is calculated by using the transfer function emulating threshold logic which is called sigmoid function.

$$z_i^n = f(u_i^n - h_i^n) \tag{2}$$

where h_i^n is the threshold value for the neuron. Usually, we use the sigmoid function for the function $f(.)$.

$$f(y) = 1/(1 + \exp(a - y)) \tag{3}$$

The weight $w_{i,j}^{n,n-1}$ and bias h_i^n are the critical parameters of a neural networks and need to be estimated during a training process which is usually based on gradient descent learning to minimize some kind of error function over a set of training observations.

The signal which is backpropagated from i th neuron in output layer N is obtained by using the difference between the output of output layer N and the prescribed observation d_i^N as follows.

$$\delta_i^N = (d_i^N - z_i^N)g(u_i^N), g(y) = f(y)(1 - f(y)) \tag{4}$$

Similarly, the signal backpropagated from the i th neuron in layer n to neurons in layer $n-1$ is given as

$$\delta_i^n = df(u_i^n)/du \sum_k \delta_k^{n+1} w_{k,i}^{n+1,n} \tag{5}$$

In summary, the update for the weight and the threshold value is given as follows where $t, t+1$ are the time step in the update.

$$\Delta w_{i,j}^{n,n-1}(t) = \eta \delta_i^n x_j^{n-1} \alpha \Delta w_{i,j}^{n,n-1}(t-1) \tag{6}$$

$$w_{i,j}^{n,n-1}(t+1) = w_{i,j}^{n,n-1}(t) + \Delta w_{i,j}^{n,n-1}(t) \tag{7}$$

$$\Delta h_i^n(t) = \eta \delta_i^n + \alpha \Delta h_i^n(t-1) \tag{8}$$

$$h_i^n(t+1) = h_i^n(t) + \Delta h_i^n(t) \tag{9}$$

where η, α are constants to confirm the convergence.

2.3. Neural network rule extraction

As universal approximators, neural networks can achieve significantly better predictive accuracy compared to models that are linear in the input variables. However, their complex mathematical internal workings prevent them from being used as effective management tools in real-life situations where besides having accurate models, explanation of the predictions being made is essential.

In the literature, the problem of explaining the neural network predictions has been tackled by techniques that extract symbolic rules from the trained networks. These neural network rule extraction techniques attempt to open the neural network black box and generate symbolic rules ith the same predictive power as the neural network itself.

In the decomposition algorithm such as the Neurorule, we start to extract rules at the level of the individual hidden and output units by analyzing the activation values, weights, and biases. Decompositional approaches then typically treat the hidden units as threshold units.

For example, the algorithm of the Neurorule is summarized as follows [4].
Step 1. Train a neural network to meet the prespecified accuracy requirement.
step 2. Remove the redundant connections in the network by pruning while maintaining its accuracy.
Step 3. Discretize the hidden unit activation values of the pruned network by clustering.
Step 4. Extract rules that describe the discretized hidden unit activation values in terms of the networks inputs.
Step 5. Generate rules describe the discretized hidden unit activation values in terms of the network inputs.
Step 6. Merge the two sets of rules generated in Step4 and 5 to obtain a set of rules that relates the inputs and outputs of the network.

Neurorule assumes the data are discretized and represented as binary inputs using the thermometer encoding for ordinary variables and dummy encoding for nominal variables.

2.4. Neural network rule extraction based on the GP

Even though the decompositional approachs such as Neurorule play a role to assess the relevant rules using the hidden layer on the basis of activation values, the procedure included in Step 5 through 6 is not straight forward.

In Neurorule, we must at first discretize the level of signal in hidden layers, and then also examine the relation between the input signal and discretized level in hidden layers. Even more, in the extraction process of rules in Trepan, we must employ the conventional method presented by Quinlann in a multiplicative way. The overall algorithm become to be complicated to obtain the expression for rules.

If the rule extraction after discretizing the signal is automatized, then it is expected that we can easily obtain relevant rules on the basis of routine works.

Therefore, we introduce the GP procedure in place of Step 4 through Step 6 in Neurorule algorithm.

In the following, we use the GP procedure to rule extraction in the Neurorule algorithm. We follow the basic procedure of Neurorule to discretize the input signal into binary representation, but we replace complicated steps to reduce the rules in Neurorule by the GP. Then, we need no steps for examining the relation between the input signal and the activation values in the hidden layers.

The overview of the GP procedure for generating rules is summarized as follows. Fig.1 shows the overview of the system.

Step 1. Discretize input variables

Similar to Neurorule, the input variables are descretized by using threshold variables and are represented in binary forms (inputs).

Step 2. Train a neural network using the discretized input variables to meet the prespecified accuracy requirement.

step 3. Remove the redundant connections in the network by pruning while maintaining its accuracy. Then, we obtain substantially important statements represented in binary forms.

Step 4. Generate rules based on the GP using the binary representation as the terminal variables (statements) in the logical expression.

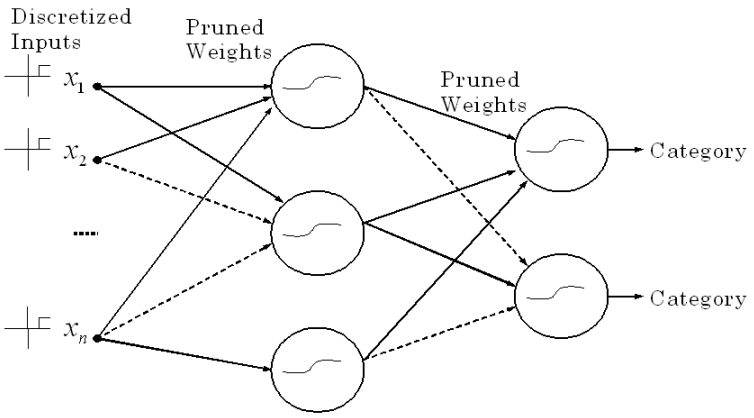


Figure 1: Overview of systems

2.5. Comparison of computational complexity

Even though we use the same discretization process for input variables as Neurorule, we find several advantages in the rule generation algorithm based on the GP used in the paper. Most of the computation time for rule generation is used for finding the discretized input variables, however, the time necessary for successive procedure Neurorule to estimate production rules is also not negligible form the aspect of computational complexity in the generation of tentative rules and combining these rules.

In the design of multi-layer neural networks, we start from an oversized networks, and then gradually remove the irrelevant connections. When all connections to a hidden neuron have been pruned, this neuron can be removed from the network. The selection of for pruning is achieved by inspecting the magnitude of their weights. A connection with sufficiently small weigh can be pruned from the network without affecting the network's classification accuracy. It is known that the optimization of parameters such as weights and threshold values in neural network by using the BP is very time consuming.

In Neurorule, once a trained and pruned network has been obtained, the activation values of all hidden neurons are clustered to simplify the rule extraction process. In the case of hyperbolic tangent hidden neurons, the activation values lie in interval [-1,1]. A greedy clustering algorithm then starts by sorting all these hidden activation values in increasing order. Adjacent values are then merged into a unique discretized value as long as the class labels of corresponding observations do not conflict. The merging process hereby first considers the pair of hidden activation values with the shortest distance in between. An equivalent process using the χ-square test statistics to merge the hidden activation values.

In Neurorule, the rule extraction process is three-hold, and is very complicated. At first, we obtain activation values of hidden unit by clustering (Step 3), and then describe tentative rules by using the activation values and outputs (Step 4). Even more, we must also generate tentative rules to describe the relation between activation values and inputs (Step 5). Then, finally we merge two corresponding rules obtained in Step 4 and 5 to generate the relations between inputs and outputs to get ultimate production rules.

We find two important drawbacks in the algorithm of Neurorules compared with the method proposed in the paper. Firstly, in the process for generating tentative rules instep 4 and 5, we can use conventional method to reduce induction rules based on the entropy. However, it is pointed out that conventional decision tree induction algorithms typically suffer from having fewer and fewer training observations available at lower levels of the tree. Neurorule utilizes similar tree induction algorithm to generate tentative rules in Step 4 and 5.

Secondly, the reduction process is mainly based on finding the relations between activation values and inputs as well as outputs. After clustering (merging) the values of hidden layer units, we must combine the inputs and outputs by considering the tentative rules which are only discriminated by the activation levels. Then, we can have no statistical measure to assess the relevancy of combinations. In contrast to these drawbacks, the GP procedure provides us a simple and comprehensive algorithm for generating rules. Once we obtain discretized inputs, we can generate production rules in a simple way using the GP which improves the fitness of rules based on the genetic operations. Even more, the relevancy of generated rules is directly obtained by observing the maximum fitness of individuals. Comparing the time necessary to optimize these parameters in neural networks, the time to execute GP procedure is very small.

Table 1 shows an example of computational complexity of Neurorule and GP method

proposed in the paper in the execution time and steps of programs. The compilation and execution of programs are done on FUJITSU VPP5000/64 UXP/V (System V) system. We also show the computational complexity of Trepan system which is worse than Neurorule, and only the performance is listed in Table 1 without explaining the details of algorithm. Then, we can compare the. capability of three systems. In the example, we assume that rules combining one logical output having six different values is approximated by using input variables which are already discretized (eight variables), and activation values (ten units in the hidden layer) are already given. We also assume that the initial population of 1000 individuals in the GP procedure is given. The program for Neurole includes the method for finding relevant clustering and combining tentative rules without human interactions, The program of the GP procedure includes initialization phase.

As is seen from Table 1. the GP method of the paper is better than Neurorule both in the complexity and computation time to obtain production rules.

Table 1: Comparison of computational complexity

	Neurorule	Trepan	GP method
Program size	1865 steps	3270 steps	385 steps
Execution Time	300	780	90

3. Production Rules and the GP

3.1. Tree representation of production rules

For simplicity, we start with the GP operations on the arithmetic expressions. The GP is an extension of the conventional GA (Genetic Algorithm) in which each individual in the population (pool of individuals) is a computer program composed of the arithmetic operations, standard mathematical expressions and variables [6][7][9]-[12][15]-[17][24].

There are several way to represent the mathematical expressions in the GP, and among them the prefix representation is approved due to the simplicity for GP operations.

The prefix representation is equivalent to the tree representation where the external points(leaves) of the tree is labeled with terminals (i.e., constants and variables), and the root of the tree is labeled with the primitive function such as binomial operation $+, -, \times, /$, and the operation taking the square root of variable. For example, we have the next prefix representation.

$$6.43(x_1 + 2.0x_2) + 3.54x_3 \rightarrow + \times 6.43 + x_1 2.0 \times x_2 \times x_3 3.54 \tag{10}$$

The prefix representation is also called as "polish notation" which is familimar to us in the design of compilers for processing programming languages. The prefix representations used in the GP procedur are called as individuals.

The equation represented by using the prefix are interpreted based upon the stack operation.

We must ensure that after initialization, crossover operation and mutation, we have a valid representation of a tree. For this purpose, the so-called stack count (denoted as *StcckCount* is useful [24]. The *StackCount* is the number of arguments it places on minus the number of arguments it takes off the stack. The cumulative *StackCount* never becomes positive until we reach the end at which point the overall sum still needs to be 1.

In the initialization phase, we generate pool of individuals used for GP operations by using the random numbers from which we define the terminal symbols and operators in

the prefix representations. Then, we can examine the syntactical validity of each individual by checking the *StackCount*. If the overall sum of the *StackCount* of generated prefix representation is not one, then the individual is not included in the pool of individuals. Even though we use such kind of cut and try method for initialization phase, the computation time for the generation of individuals is small enough.

By using the *stackCount* already mentioned, we can know the terminal of the subtree which is the candidate for the crossover operation. The basic rule is that any two loci on the two parents genomes can serve as crossover points as long as the ongoing *StackCount* just before those points is the same. The crossover operation creates new offsprings by exchanging sub-trees between two parents.

Before applying the genetic operation, we must evaluate the ability of each individual (tree structure). The ability is called as the fitness, and is calculated based upon the comparison between the predicted value by the individual and the observed value. We assume that we have a set of input variables $x_1, x_2, ..., x_n$ and output value y for multiple observations, say 100 observations. Then, we approximate the function $F(x_1, x_2, ..., x_n)$ so that the value of function $F(.)$ becomes close to y for each observation. In the GP method, the approximation of function is represented in prefix representation corresponding to an individual. Then, we have the root mean square error ($rmse$) between $F(x_1, x_2, ..., x_n)$ and y for each individual. If the $rmse$ obtained by a individual is small, then the ability of the individual to approximate the input-output pair is seemed to be high. Therefore, we define the ability (called as fitness) of each individual as the inverse of $rmse$. In the GP procedure, the individual having higher fitness can get higher probability used for generating offsprings (children) , and can survive keeping the effect longer in the pool of individuals.

We iteratively perform the following steps until the termination criterion has been satisfied.

(Step 1)

Generate an initial population of random composition of the functions and the terminals of the problem (constants and variables).

(Step 2)

Execute each program (evaluation of system equation) in the population and assign it a fitness value using the fitness measure. Then, sort the individuals according to the fitness S_i.

(Step 3)

Create a new population of computer programs by applying the following two primary operations. The operations are applied to the individuals chosen with a probability $p_i$based on the fitness. The probability p_i is defined for ith individual as follows.

$$p_i = (S_i - S_{min}) / \sum_{i=1}^{N} (S_i - S_{min}) \tag{11}$$

where S_{min} is the minimum value of S_i, and N is the population size.

Create new individuals (offsprings) from two existing ones by genetically recombining randomly chosen parts of two existing individuals using the crossover operation applied at a randomly chosen crossover point.

(Step 5)

If the result designation is obtained by the GP (the maximum value of the fitness become larger than the prescribed value), then terminate the algorithm, otherwise go to Step 2.

We apply the mutation operations defined as follows at a probability p_M. Even though the crossover operation can generate various kinds of offsprings, but sometime we find cases

where very similar individuals occupy the whole members of the pool. Then, the crossover operation is not usable any more to generate different kinds of individuals. Then, we apply the mutation operations in the GP procedure.

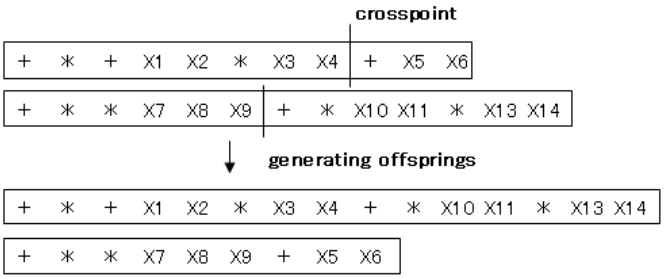
The goal of the mutation operation is the reintroduction of some diversity in an population of individuals. Two types of mutation operation in GP is utilized to replace a part of the tree by another element. We select a individual I_s at random from the pool. In the first type of mutation (called global mutation), we apply the crossover operation to I_s for which we use dummy individual as the pair of crossover operation. Then, the overall structure of I_s is changed. In the second type of mutation (called as local mutation), we replace only a part of I_s at random, then the individual I_s is slightly changed. The overview of these mutation operations are also shown in Figure 2.

(Global mutation)

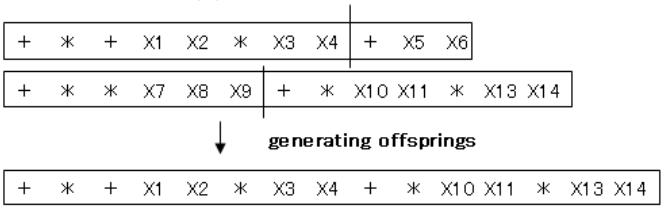
Generate a individual I_s, and select a subtree which satisfies the consistency of prefix representation. Then, select at random a leaf in the individual to which the mutation is applied, and replace the leaf by the subtree of the individual I_s.

(Local mutation)

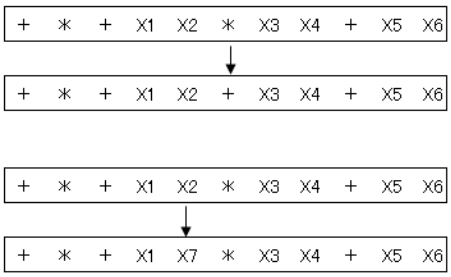
Select at random a leaf in the individual to which the mutation is applied, namely, we replace the parameter in the leaf by another value (a primitive function or a variable).



(a) crossover operater



(b) Global Mutation for primitive function



(c) Local Mutation for certain variable

Figure 2: Overview of GP operations

3.2. Applying the GP to rule generation

We can define the condition part of a rule as a logical expression (formula) which is represented as a connection of propositions with logical operators including AND, OR ,for example we have

```
if v1 > (0.2 *v2+2.0 *v3) AND (v4 <0.4)   then class=A        (12)
```

where $v_1, v_2, ...$ are input variables for the systems. A proposition can be defined as a connection of two arithmetic expressions (equations) with operators, including $>, \geq, <, \leq, =, \neq$. Originally, the arithmetic expressions in propositions should be generated by the GP procedure used for the arithmetic expressions. However, the straightforward application of GP to the generation of proposition is very complicated. Then, the rule extraction using the neural networks helps us to avoid the proposition generation.

As previously described, in our system based on the neural networks, the input variables are discretized by using threshold values. Even more, by pruning the connections (weights) in neural networks, we have finally several binary expressions used for the input to the simplified neural networks. Then, these binary expressions are used as the propositions included in the logical expressions in the GP. For example, we define new logical variables x_i represented by input variable v_i such as

$$X_{kj} = \begin{cases} True, & \text{if } v_i = j; \\ False, & \text{otherwise} \end{cases} \tag{13}$$

where a_i is a constant reduced from selecting of threshold values.

Then, we find that the logical expressions included in the production rules are the same as the arithmetic expression using prefix representation by replacing the operands by the propositions, and the arithmetic operators by the logical operators. For example,
numerical variables $x_i \rightarrow$ logical variables x_i
arithmetic operators $+, \times \rightarrow$ OR, AND

We also define the fitness of individuals as the accuracy of rule generated by the rule corresponds to the underlying individual. To improve the fitness of individuals, we apply the GP operations to the logical expressions.

The fitness of individuals is evaluated as follows.
(1)calculation of arithmetic expressions.

By substituting the value of input variables v_i, we can evaluate the binary arithmetic expressions included in propositions such as equation (10). For example, a set of specific values for $v_1, v_2, ..., v_n$ are used to obtain numerical values.
(2)interpretation of propositions.

Since we know the values of arithmetic expressions, the logical value of logical variables x_i are obtained by combining the propositions and comparative operators such as equation (13).
(3)interpretation of logical formula.

Finally, we can know the logical value of the whole logical formula (individual) by applying the logical operations among propositions. As already mentioned, the value is compared with the prescribed observation r to calculate the fitness.

4. Applications

4.1. Prediction of bankruptcy

There have been a fair number of previous studies for predicting corporate failure (bankruptcy) using probabilistic estimate such as logit model, multi-variate discriminant analysis (MDA),

and neural networks. Among them, the MDA and similar approaches have been the most popular for bankruptcy studies based on the vector of financial ratios reflecting financial status of firms [10][11].

In this section, we apply the neural network rule extraction based on the GP method proposed in the paper to derive the linguistic rules to predict the bankruptcy.

The collection of data for bankruptcy firms requires a definition of failure and specification of firms. The definition of bankruptcy in this section is defined as the occurrence of defaults. The population are restricted by (i) the period from 1970 to 1986 in Japanese industries, (ii)the equity of the company had to have been traded on some stock market to exclude small firms, (iii) the company must be classified as an industrial firm.

These procedure are used to generate a list of failed firms satisfying the inclusion criteria.

At the next step, we must also gather the data for non-bankruptcy (sound) firms to specify the binary sample space as well as in ordinary MDA. Our two samples of firms consist of 26 bankrupt firms and matched samples of 26 non-bankruptcy entities. The latter are matched to the failed group by industry and year of the data. The selection of non-bankruptcy firm must reflect the binary feature, then we select a pair of failed and sound firm in the same industry and are resemble in size of firms. The pair of firms should exist in the fiscal year just before the bankruptcy. Among 29 firms, 19 firms in each group are selected as learning samples, and the rest firms are used for testing.

The next phase is one of actually collecting financial data for the bankruptcy firms. Each report has to include the balance sheet, income statement, fund statement. We have at first prepared 62 financial ratios (vector of input variables) for the prediction of bankruptcy. However, it is noted that a number of variables obey very deformed distribution among firms, and are seemed to be irrelevant for statistical inference. Then, these variable are removed from the pool of input variables based on the T-test and Wilcoxson's rank test for normal distribution.

Then, we have 13 variables to predict the bankruptcy, but the number is relatively large to derive concise prediction rule in an linguistic manner. Therefore, we applied the principal component analysis to these 13 variables, and determined 5 variables for prediction.

In the preprocessing by the neural network to discretize the input variables, we use four threshold value for each financial ratios (variables). Then, we apply the binary prediction for learning using the backpropagation to derive the meaningful propositions included in the production rules. Then, we find that according to the pruning algorithm, 13 binary input are pruned, leaving only 5 binary inputs in the neural network. This corresponds to 5 of the original inputs.

The conditions for simulation studies are summarized as follows.
Number of layers of neural network:three
Number of units in hidden layer:10
Number of output units:two
Maximum size of array in individuals (denoted as M_s)=10
Population of individuals:1000
GP generations for learning:100

Definitions for input variables ultimately used for classifications are as follows.

x1:return on capital employed, x2:turnover ratio of fixed assets, x3:ordinary ratio of profit to net sales, x4:current ratio, x5:debt ratio.

In Table 2, we show the result for classification by using four individuals included in the pool with respect to the values of fitness. In Table 2, we also show the result obtained by MDA(Multivariate Discriminate Analysis) and P-NN(Pruned Neural Network). Different

from the direct tree growing method such as C4.5, the GP method provides us various tree structures corresponding to individuals, then we use several important rules preserved in the pool by considering the value of fitness.

Table 3 summarizes the explanatory (linguistic) rules to predict the bankruptcy. In Table 3, we show top four individuals included in the pool with respect to the values of fitness. These four individuals can cover all of the explanatory classification for failed firms. As is seen from the table, the explanation included in rules is understandable for user to recognize the reason of bankruptcy, while the maximum size of array in individuals is limited to a small value (maximum size $M_s = 10$). If we increase the maximum size to larger value such as $M_s = 20$, then the true classification becomes to be very close to 100 %, even though the simplicity of explanation is lost (details are omitted here).

Table 2: Prediction of bankruptcy(true classification rate %)

category	GP method	P-NN	MDA
failed	86	83	82
sound	88	81	78

Table 3: Examples of generated rules

Or And X3 X1 Or X5 X4
Or X4 Or Or X1 X4 And X5 X4
Or X1 Or X4 Or X2 X4
Or X5 Or X4 And X3 X4

4.2. Applications to German credit data

The experiment on real-life credit-risk evaluation is carried out using the German credit data by showing comparative study with Neurorule. The German credit data is obtainable from the website(URL: http://www.liacc.up.pt/ML/statlog/datasets/german/german.doc.html). The data consist of 1000 records of personal loan, and the input variables for one record include 7 numerical data and 13 categorical data.

By using the discretization to numerical and categorical data, and then we split at random the data into then use two-third training data set and one-third test set. When representing all discretized inputs using the thermometers and dummy encoding, we ended up with 45 binary inputs. Note that according to the pruning algorithm, 37 binary input are pruned, leaving only 8 binary inputs in the neural network. This corresponds to 6 of the original inputs.

Because our main purpose is to develop intelligent credit-risk evaluation system that are both comprehensive and user friendly, it is obvious that simple, concise rule sets are to be preffered. Hence, we will also take into account the complexity of the generated rules or trees as the performance measure.

The complexity will be quantified by looking at the number of nodes including terminal nodes and intermediate nodes.

As in shown in reference [1], for the German credit data set, Neurorule yielded a highest test set classification accuracy than C4.5 rules and extracted only 4 propositional rules. The number of rules is very compact compared to C4.5 and Trepan systems.

The conditions for simulation studies are summarized as follows.
Number of layers of neural network:three

Number of units in hidden layer:55
Number of output units:two
Maximum size of array in individuals:$M_s = 10$ and 20
Population of individuals:1000
GP generations for learning:100 (for $M_s = 10$) and 200 (for $M_s = 20$)

Definitions for input variables ultimately used for classifications are as follows.

x1:checking accout(4 nominal), x2:duration in month(numerical), x3:credit history(5 nominal),x4:purpose(11 nominal), x5:saving accout(5 nominal),x6:other parties(3 nominal).

These results cited from Reference [4] and our result for simulation studies are summarized in Table 4 and Table 5. Table 4 shows the comparison of mean true classification rate, and Table 5 describes the comparison of complexity of rules for each system. In these tables, P-NN means the method using pruned neural networks, and GP-1 means the GP method of our paper where the maximum size of array is $M_s = 10$, and GP-2 means the cases for $M_s = 20$. In our method, we show the result for classification by using four individuals included in the pool with respect to the values of fitness.

In Table 6, we show top four individuals included in the pool with respect to the values of fitness in cases for $M_s = 10$. These four individuals can cover all of the explanatory classification for bad (denied) applicant classification.

As is seen from Table 4, true classification rate of our system is comparable to Neurorule in the mean, if we select the maximum size of array to be $M_s = 10$. If we increase the maximum size to larger value such as $M_s = 20$, then the true classification becomes to be better, even though the complexity of explanation increases as shown in Table 5.

The result in Table 5 suggests us that the explanation included in rules is understandable for user to recognize the reason of bad applicant while the maximum size of array in individuals is limited to a small value ($M_s = 10$).

Table 4: Result of inference (mean true classification rate in %)

C4.5	P-NN	Neurorule	Trepan	Nefclass	GP-1	GP-2
74.25	77.84	77.25	73.95	73.65	77.23	79.98

Table 5: Comparison of rule complexity

methods	complexity
C4.5	17 propositional rules
P-NN	6 inputs
Neurorule	4 propositional rules
Trepan	11 leaves, 21 nodes
Nefclass	14 fuzzy rules
GP-1	4 propositional rules(4 leaves,3 nodes)
GP-2	4 propositional rules(8 leaves,7 nodes)

4.3. Application to bond rating

Industrial corporate bonds have been assigned quality rating since the early 1990s. Each year private organizations such as Moody's and Standard & Poors assign to a portion of new bond issues that year to provide investors with a simple system of guarantee for relative investment qualities. Bond ratings attempt to provide a simple measure of the

Table 6: Examples of generated rules

Or And X4 X1 Or X6 X4
Or X5 Or Or X1 X4 And X6 X4
Or X1 Or X4 Or X3 X5
Or X6 Or X5 And X4 X2

relative investment quality of these securities. A rating simply helps investors determine the relative likelihood they might lose money on a given fixed income investment. Under present commercial bank regulations only bonds rated in four classification are eligible such as Aa, A, Ba Ba or B.

A number of previous studies on bond rating attempt to predict corporate bond ratings based on the financial characteristics for firms issuing the bonds. The MDA is also used for the rating where several (four or five) financial ratios are used for the discriminant variables.

Then, the neural network rule extraction based on the GP proposed in the paper is applied to derive linguistic rules so that the inference using the input variables is available to discuss the credit worthiness of firms.

At first, we select three groups of Japanese companies. The information of rating for the companies are obtained from the published data of Moodys' Japan in 2000. Even though, the range of rating varies from AAA to CCC, usually it is very rare to categorize the company to the lower ranks such as C or CC. The Japanese companies treated here are relatively good, and the range of rating is included in AAA through CCC. The output (rating) is categorized into these three categories.
1)30 companies ranging from Aaa1 through A3 in Moody's rating are merged into a single category (category A)
2)32 companies ranging from Baa1 through B3 in Moody's rating are merged into a single category (category B)
3)27 ranging from Caa1 through C3 in Moody's rating are merged into a single category (category C)

We select these companies from three industries so that we can prove the robustness of the rule. It is observed that the distribution of the financial ratio between another industries are usually different. Each dataset is randomly split into two-third training data set and one-third test set in each category.

We use ordinary financial ratios (real numbers calculated by using the financial statement yearly published by the Japanese companies) as the input variables to the rules.

Before utilizing these financial ratios, we must examine the statistical proportions of the financial ratios. If the distribution of a financial ratio is far apart from the normal distribution, then the financial ratio is removed from the candidate of the input variables for the system.

Furthermore, by using the distribution of financial ratio, we remove insignificant sample of financial ratio from the pool which is located outside of the 5 σ of the distribution. Finally, we obtain 24 input variables for original neural network inputs.

Then, we apply the discretization for input variables to make binary inputs. After discretization, we use the procedure to prune the weight of the neural networks. Note that according to the pruning algorithm, 24 binary input are pruned, leaving only 9 binary inputs in the neural network. This corresponds to 9 of the original inputs.

In the following, we evaluated the inference by the rules generated by the GP. The ability of a rule is calculated by the number of samples where the rating given by the system and

the rating by the rating agency are the same (the same interpretation).

The conditions for simulation studies are summarized as follows.
Number of layers of neural network:three
Number of units in hidden layer:35
Number of output units:three
Maximum size of array in individuals:$M_s = 12$
Population of individuals:1000
GP generations for learning:100

Definitions for input variables ultimately used for classifications are as follows.

x1:operating profit ratio of equity capital, x2:return on equity capital(on after-tax basis), x3:turnover ratio of fixed asset, x4:return on capital employed, x5:increased receipts ratio, x6:rate of equity capital growth, x7:liabilities ratio of operating cash flow, x8:current liabilities ratio of operating cash flow, x9:return on stockholder's equity.

Table 4 shows the comparison of mean true classification rate, and Table 5 describes the comparison of complexity of rules for each system. In these tables, P-NN means the method using pruned neural networks, and GP means the GP method of our paper where the maximum size of array is $M_s = 20$.

Table 7: Prediction of bond rating(mean true classification rate in %)

category	GP method	NN	MDA
A	83.33	86.66	76.66
B	81.18	84.84	72.72
C	81.48	85.18	70.37

Table 8: Examples of generated rules

A	Or X8 Or Or X2 Or X6 X2 X3
	Or Or X3 Or X2 Or And X5 Or X9 X2 X6 X1
	Or Or X5 X2 Or Or X7 And X4 X9
	Or Or Or X4 X6 Or X6 X2 And And X9 X5 X3
B	Or Or X5 X2 Or Or X7 And X4 X9
	Or X8 Or Or X2 Or X6 X2 X3
	Or Or Or X4 X6 Or X6 X2 And And X9 X5 X3
	Or Or X3 Or X2 Or And X5 Or X9 X2 X6 X1
C	Or Or X6 And And X8 X6 X5 Or X7
	Or X8 Or Or X2 Or X6 X2 X3
	Or Or X5 X2 Or Or X7 And X4 X9 X7
	Or Or X1 Or X5 X4 Or X4 X7

Table 7 describes the mean true classification rate for each category compared to the result obtained by neural network methods and MDA. We show the result for classification of our method by using four individuals included in the pool with respect to the values of fitness. In Table 8, we show individuals included in the pool having highest fitness. Including other three individuals, these individuals can cover all of the explanatory classification for bad (denied) applicant classification.

As is seen from Table 7, true classification rate of our system is placed between NN and MDA in the mean, if we select the maximum size of array to be $M_s = 20$. Since the category for explanatory clasification is three, and the number of substantial input variable necessary

for classification is relatively large compared to previous two applications, explanatory rules becomes to be slightly complicated.

5. Conclusion

This paper treated the use of neural network rule extraction techniques based on the Genetic Programming (GP). We used the neural network and binary classification to obtain simple and relevant classification rules by pruning weight among neurons to obtain simple but substantial binary expressions which are used as statements is classification rules generated by the GP. The method was applied to extract rules to prediction of bankruptcy and the classification of corporate bonds (rating) by using the financial indicators.

For further works, there exist the extension of the method to various rule extraction problems and to slightly complicated functional forms rather than fundamental arithmetic. Further works will be done by the authors.

References

[1] F. Allen and R. Karjalainen: Using genetic programming to find technical trading rules. *Technical Report* (Wharton School, University of Pennsylvania, 1993).

[2] E.I. Altman: *Corporate Bankruptcy in America* (D. C. Health and Company, 1971).

[3] R. Andrews, J. Diederich, and A.B. Tickle: A survey and critique of techniques for extracting rules from trained neural networks. *Knowledge Based Systems*, **8-6** (1995), 373-389.

[4] B. Baesens, R. Setiono, C. Mues, and J. Vanthienen: Using neural network rule extraction and decision tables for credit-risk evaluation. *Management Science*, **49-3** (2003), 313-329.

[5] C.M. Bishop: *Neural Networks for Pattern Recognition* (Oxford University Press, U.K., 1995).

[6] X. Chen and S. Tokinaga: Approximation of chaotic dynamics for input pricing at service facilities based on the GP and the control of chaos. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E85-A-9** (2002), 2107-2117.

[7] X. Chen and S. Tokinaga: Synthesis of multi-agent systems based on the co-evolutionary genetic Programming and its applications to the analysis of artificial markets. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences (Japanese Edition)* , **E86-A-10** (2003), 1038-1048.

[8] M.W. Craven and J.W. Shavlik: Extracting tree-structured representations of trained networks. In D. Touretzky, M. Mozer, and M. Ha- sselmo (eds.): *Advances in Neural Information Processing Systems (NIPS)* (MIT Press, Cambridge, MA), **8**, 24-30.

[9] Y. Ikeda: Estimation of the chaotic ordinary differential equations by co-evolutionary genetic programming. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E85-A-4** (2002), 424-433.

[10] Y. Ikeda and S. Tokinaga: Approximation of chaotic dynamics by using smaller number of data based upon the genetic programming. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E83-A-8** (2000), 1599-1607.

[11] Y. Ikeda and S. Tokinaga: Controlling the chaotic dynamics by using approximated system equations obtained by the Genetic Programming. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E84-A-9** (2001), 2118-2127.

[12] Y. Ikeda and S. Tokinaga: Chaoticity and fractality analysis of an artificial stock market by the multi-agent systems based on the co-evolutionary Genetic Programming. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E87-A-9** (2004), 2387-2394.

[13] Y. Ikeda, S. Tokinaga, and J. Lu: Neural network rule extraction using Genetic Programming and its applications. *Proceedings of 2004 International Symposium on Nonlinear Theory and its Applications* (2004), 485-488.

[14] M.J. Keith and M.C. Martin: Genetic programming in C++: Implementation issues. In K.E. Kinnerar, Jr. (eds.): *Advance in Genetic Programming* (MIT Press, 1994).

[15] J.R. Koza: Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. *Report No.STAN-CS-90-1314* (Dept.of.Computer Science Stanford University, 1990).

[16] J.R. Koza: Evalution and subsumption using genetic programming. *Proceedings of the First European Conference on Artificial Life* (MIT Press, 1991).

[17] J.R. Koza: *Genetic Programming* (MIT Press, 1992).

[18] B. Lev: *Financial Stament Analysis* (Prentice-Hall, 1974).

[19] N. Nauck: Data analysis with neuro-fuzzy methods. *Habilitation Thesis* (University of Magdeburg, 2000).

[20] M. Oussaidene, B. Chopard, O.V. Pictet, and M. Tomassini: Parallel genetic programming and its application to trading model induction. *Parallel Computing*, **23** (1997), 1183-1198.

[21] Y.H. Pao: *Adaptive Pattern Recognition and Neural Networks* (Addison-Wesley Publishing Co., Inc., 1989).

[22] J.R. Quinlan: *C4.5 Programs for Machine Learning* (Morgan Kafumann, Chambery, France, 1993).

[23] K. Tan and S. Tokinaga: The design of multi-stage multi-stage fuzzy inference system with smaller number of rules based upon the optimization of rules by using the GA. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, **E82-A-9** (1999), 1865-1873.

[24] M. Yababe and S. Tokinaga: Applying the genetic Programming to modeling of diffusion processes by using the CNN and its applications to the synchronization. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences (Japanese Edition)*, **J85-A-5** (2002), 548-559.

Shozo Tokinaga
Graduate School of Economics
Kyushu University
6-19-1 Hakozaki, Higashi-ku
Fukuoka 812-8581, Japan
E-mail: tokinaga@en.kyushu-u.ac.jp