Generating the Production Rules by using the Genetic Programming and its Applications to Trading and Rating Model Induction

Matsuno, Seigo

https://doi.org/10.15017/3000302

出版情報：経済論究. 112, pp.181-197, 2002-03-29. 九州大学大学院経済学会
バージョン：
権利関係：

Generating the Production Rules by using the Genetic Programming and its Applications to Trading and Rating Model Induction

Seigo Matsuno

1 Introduction

This paper deals with the estimation of production rules by using the Genetic Programming (GP) and its applications. It is an important task to find out the rules and decision tree based on the observation, especially to avoid the risk in business activities. There exist several method to generate production rules by using theoretical methods such as the entropy-based decision tree (for example, ID3) and statistical procedure to generate linguistic if-then rules (inductive learning) [1] [2]. However, the preprocessing of the observation is completely separated from the estimation of rules, and the facts limit the applicability of the method. Moreover, the straightforward application of these theoretical methods lead us to very complicated rules, and they are not more understandable for users.

In previous works, we demonstrated that the GP is capable of estimating the system dynamics by using the observation (time series) which is generated by a deterministic equations such as the chaotic dynamics [9] [10]. Moreover, a straightforward application of estimated equations lead us a stable control method of a chaotic dynamics by imposing a small input to the system. These GP procedures are able to be extended to the generation of production rules by changing the arithmetic operations to logical operations.

Authors show several application of the GP the induction of trading model for the stock market [3]. The parallel genetic programming is utilized to improve the induction of trading rules, and specific data structure is proposed [4] [5]. However, the paper emphasize the computation time, and the logical expressions included in the rule are very simply, and very hard to extend to more general cases.

The GP used in the paper generate production rules by combing the optimization of arithmetic expressions included in the rules [6]. At the same time, by controlling the complexity of rules, and finally we can obtain reasonable interpretation and decision tree [6]. Originally, the if-then rules are represented in tree structure, but it is not relevant form for applying the genetic operations. Then, we represent the tree in a string (individual) by using the prefix representation to treat the nesting structure of production rules.

The genetic operation for the generation of production rules is applied at first to the logical

formula itself. But, the method is extended so that the lower level elements such as the arithmetic expressions included in propositions. In the crossover operations, the crosspoints are basically randomly selected, but to keep the consistency of operations, a kind of counter is introduced. The GP operations for these procedure are consistent and simple enough to apply the method to various field of decision making.

Then, we apply the GP method to generate the rules to to predict the stock prices. As another applications, we generate rules to 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, the relation between the production rule and the GP is described. 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 trading model induction using the stock prices. Section 5 describes the application to the rating rule induction, and in Section 6, we compare there result with that of conventional inference systems.

2 Production rules and the GP

2.1 Tree representation of production rules

The production rule is a well known form to describe the inference process, and generally represented as

if A then B

where A is an antecedent (condition) and B is a consequence (result) of the rule. It means that if the condition A is satisfied, then the result B is employed. The result B states some conclusion such as the rise of a stock price. The result B can have several cases for consequences, but in the paper, we restrict ourselves to one consequence. Then, several rules are generated simultaneously, for example, we have two rules if the underlying observation belongs to Category 1 and Category 2. As another example, in the prediction of stock price, the antecedents include description about the feature of the stock price, and the consequence is used to predict whether the future stock price rise. One another rule is prepared for the fall of price.

Therefore, the condition A become to be a simple logical formula L. The formula is composed of several propositions P_j and logical operators O_k. In general, a logical formula is represented by combining simultaneously several proposition and logical operators. But, to simplify the GP operation for generating the production rules, we consider the case where the number of proposition combined with a logical operator in a formula is restricted to two. By using the tree structure, it is represented by a root denoting the logical operator, and left and right branches as

two propositions accompanied. By arranging these subtrees, we have a whole tree structure representing the production rule as shown in Fig.1

Each proposition is composed of a single relative operator R_i and two distinct arithmetic expressions A_k, A_{k+1} as shown in Fig.1. For example, we have a rule denoting

if $x_1>1$ and $x_2<10$ then future price will rise

where x_1 and x_2 are input variables describing the feature of a stock price.

In the following section, the tree representations are translated to equivalent prefix representations.

2.2 Function and terminal sets in rules

A GP can be regarded as a Lisp function or a S-expression [7] [8]. It is usually represented by a parse tree. The number of nodes in the parse tree gives the measure of the space complexity of that GP.

For the description of production rules, terminal and non-terminal codes are randomly taken from some well defined terminal and non-terminal sets. To ensure the syntax validity, and the control of complexity of the GP, some restrictions must be imposed.

In the rules, not only ordinary logical operators such as AND, OR, we also use primitive operations for comparing the value of variables. Moreover, for the terminal elements, we also assume input variables and constants. In the following, the function set F and the terminal set T are defined as follows.

$$F=+,\ -,\ \times,\ AND,\ OR,\ f>,\ f=,\ f< \tag{1}$$

$$T=x_j,\ C_k,\ 0,\ 1 \tag{2}$$

where, the symbols $f>$, $f=$, $f<$ represent the relation operation for two variables. The symbol x_j denotes the j th input variable to the system, and C_k is the k th constant used for comparison, and generated by a random number. For example, $f>(x,c)$ is one (zero) if $x>c(x\leq c)$.

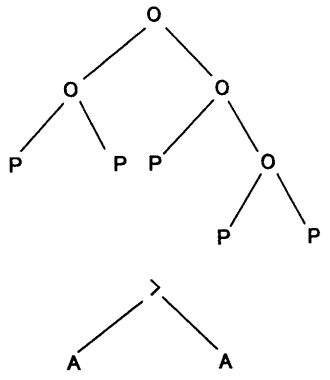


Fig.1-Tree structure for production rules.

It is also possible to extend the conditional clause other than $f>$, $f=$, $f<$ operators into functional form including arithmetic expression such as $x_1 * x_2 - x_3 > 9.5$. To simplify the algorithm, all of the functions are assumed to be binomial operations. As is seen from Fig.1, the leaf of the tree is composed of the subtree denoting the comparison of two arithmetic expressions. The functional form for these arithmetic operations can be optimized by using the GP procedure as well as the estimation of chaotic dynamics. But simultaneously, we restrict the complexity of the functional form so that the generated rule are understandable.

Followings are assumed in the paper.

(1) logical formula.

The number of operands (proposition) combined with each logical operator in a logical formula is restricted to two.

(2) prefix representation.

Each logical formula and arithmetic expression is denoted in a prefix representation.

From the definition, the number of arithmetic expressions included in a proposition is two which are combined by a relation operator.

2.3 Prefix representations

The GP is an extension of the conventional GA in which each individual in the population (pool of individuals) is a computer program composed of the arithmetic operations, standard mathematical expressions and variables [7] -[10].

We must at first design the interpreter of the genetic programming (genome interpreter) in an efficient way. The interpreter specifies the following lower level aspects of the design, namely, the representation of the nodes and the tree, the evaluation of the nodes and the whole tree, and the genetic operation on the nodes and the whole tree. The prefix representation is approved in [8] based upon the comparative study with other representation such as pointer based implementation and the postfix approach. Therefore, we use the prefix representation as the method to represent the trees which corresponds to rules.

For simplicity, at first we focus on the arithmetic expressions in the GP. In the parse tree the node non-terminal are taken from some well-defined functions such as binomial operation $+, -, \times, /$, and the operation taking the square root of variable. Terminal nodes consist of arguments chosen from set of constant and variable. The pool of variable consists of the variable $x(t)$ itself and the time lag of $x(t)$ such as $x(t-1)$.

The prefix representation follows traditional representation by using the Lisp syntax. For example, we have the next prefix representation. $[6.43 \times x(t-1) - x(t-2)] \times [x(t-3) - 3.54]$

$$\rightarrow \times - \times 6.43 x(t-1) x(t-2) - x(t-3) 3.54 \quad (3)$$

The evaluation of prefix representation is done with the stack operation. We begin to scan the

prefix representation, and if we meet the terminal (operand) then we push down the term into the stack. If we meet the node (operator), then we take out the operands from the stack, and execute the operation. The result of the operation is also pushed down to the stack.

For checking the validity of underlying parse tree, the so-called stack count (denoted as *StcckCount*} in the paper) is useful [8]. The *StackCount* is the number of arguments it places on minus the number of arguments it takes off from the stack. The cumulative *StackCount* never becomes positive until we reach the end at which point the overall sum still needs to be 1.

By using the *StackCount* we can see which loci on the prefix expression is available to cut off the tree for the crossover operation, and we can validate whether the mutation operation is allowed.

If final count is 1, then the prefix representation (tree) corresponds properly to a system equation. Otherwise, the tree structure is not relevant to represent the equation.

We can have the same style of prefix representation for the whole logical formula by replacing arithmetic operators by logical operators and variables by propositions (details are omitted here).

2.4 Improvement of rules by evaluating fitness

In the GP, the production rules are represented in the tree structure (called individuals as well as in the Genetic Algorithm :GA). One tree corresponds to a system of production rules. In the GP, the performance of each individual (called as fitness) is evaluated by comparing the output generated by the rules corresponds to the individual with the observed data to be approximated. After sorting the individuals according to the fitness, we apply the crossover operations to the individuals possessing relatively higher fitness.

The fitness of the GP is defined depending on the task of the problem. For the prediction of the times series, the output of the rule is compared with the observation so that the GP generate better rules for the prediction (learning process).

In learning process, at first we generate a pool of individuals by using the random numbers. The initial fitness is computed for each individual. Then, we apply the genetic operations (crossover operations and mutation operations) to the individuals by using the fitness, and then generate offsprings. By replacing the individuals with lower fitness by these offsprings, we have finally improved production rules.

It is assumed that we have sufficient data for learning to generate production rules. For example, we have a storage of observation of features x_1, x_2, ..., x_n for a stock price, and the result of r (rise/fall) of the stock price. The fitness of an individual is evaluated by applying the rule corresponds to the individual for the learning data composed of the set of x_1, x_2, ..., x_n and r in this example. If the conclusion of the rules is the same as the learning data, then the fitness of the individual is increased.

More precisely, the process for calculating the fitness is divided in three steps.

(1) calculation of arithmetic expressions.

By substituting the value of variables, we can evaluate the arithmetic expressions included in propositions. For example, a set of specific values for x_1, x_2,...,x_n are used to obtain numerical values.

(2) interpretation of propositions.

Since we know the values of arithmetic expressions, the logical value of each proposition is obtained by combining the propositions and relative operators.

(3) interpretation of logical formula.

Finally, we can know the logical value of the whole logical formula (individual) by applying the logical operations among proposiiotns.

As already mentioned, the value is compared with the prescribed observation r to calculate the fitness.

2.5 Data structure and input variables

While the above observation allows one to use a hierarchical data structure for representing the individuals. A natural way to embed the original parse tree into a array in memory is to divide the data for the logical formula and the proposition.

In the parse tree of logical formula, all nodes are logical operators and terminators are proposition. We call the array for the logical formula as Level-1 individuals. The array is represented as shown in schematic diagram Fig.2.

In the same way, we prepare a set of array representing the arithmetic expressions for propositions which is also shown in Fig.2 (denoted as Level-2 individuals). The data for each proposition in Level-1 is linked with corresponding data in Level-2.

Since a proposition in a logical formula is accompanied with two arithmetic expressions, the linkage is reached to two arithmetic expressions in Level-2.

In terms of input variables, in our system, the feature describing the observation is usually used to improve the efficiency of analysis, and to realize sophisticated systems. For the analysis of the times series, we use the wavelet transform for the feature extraction [3]. For the cross section analysis such as the categorization and classification of observation, we use the principal component as the input variables rather than straightforward utilization of variables.

These variables obtained by pre-processing of the observation are used as the variables in the production rules. Namely, the condition clause in equation (1) is composed of the input variables and the function sets.

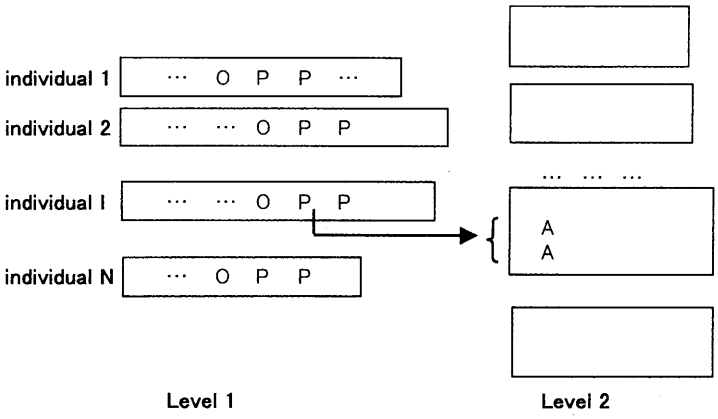


Fig.2-Level-1 and Level-2 data structure

3 Applying the GP to rule generation

3.1 Genetic operations

Two genetic operations are used in the reproduction phase: crossover and mutation. In the GP, the crossover operation is applied to a pair of individuals I_A and I_B representing the production rules from the population. The tree structure of a rules (individual) is represented as a string of symbols for function set and terminal set. Then, we select a random place L_A (called crosspoint) as a crossover operation on string I_A, and then determine the crosspoint L_B for another string I_B so that the interchange of two parts of the string lead to relevant functional forms whose operators and operands are necessary and sufficient for the arithmetic. The resulting individuals are syntactically valid and only the maximum depth constraints is checked for non-violation.

However, other than the GP procedure for the approximation of functional forms, we must carefully choose the loci of the crossover operations depending on the symbol at the crosspoint.

To realize these genetic operations, we consider following two cases for crossover operation.

Suppose we have two individuals in Level 1 denoted as R_1, R_2, as well as two individuals in Level 2 denoted as S_1, S_2. Basically, the crossover operation means the selection of random crosspoints A on R_1 and S_1 where the individuals are cut, and the replacement of of these latter halves with corresponding parts inculded in R_2 and S_2. The crosspoint B on the individuals R_2 and S_2 are also selected so that the consistency of the offsprings are kept by using the *StackCount*. In general, we find several crosspoint B in the individual, then select one of them at random.

Depending on the symbol S on the crosspoint A, we have following two scheme of crossover operations.

(CR-A) S is a logical operator

In this case, we swap the individuals R_1 and R_2 by cutting these individuals at crosspoints A and B, and by generating offsprings. In this case, the corresponding information stored in Level 2 is also exchanged.

(CR-B) S is a proposition

Then, we apply one of following three crossover operations depending on a prescribed probability.

(CR-B-1) replacement of arithmetic expressions.

In this case, we replace the arithmetic expressions included in underlying propositions for crossover operations. Suppose that the porposition P_A in individual R_1 is placed on the crosspoint A, and the proposition includes two arithmetic expressions E_1, E_2. In the same way, the individual R_2 has the proposition P_B at the corsspoint B, and the proposition includes two arithmetic expressions G_1, G_2. Then, we replace E_1, E_2 with G_1, G_2.

(CR-B-2) replacement of parts of arithmetic expressions.

In this case, we replace the arithmetic expressions included in underlying propositions for crossover operations as well as the procedure CR-B-1. However, in this case we swap only one arithmetic expression each other. Namely, only the arithmetic expressions E_1 and G_1 are replaced each other in CR-B-1.

(CR-B-3) apply the GP to arithmetic expression.

In this case, we apply the GP to the arithmetic expression among E_1 and G_2 in a similar fashion used for the approximation of equations. A random crosspoint on E_1 is chosen, and then another crosspoint is selected on G_1 by considering the consistency of offsprings based on the *StackCount*.

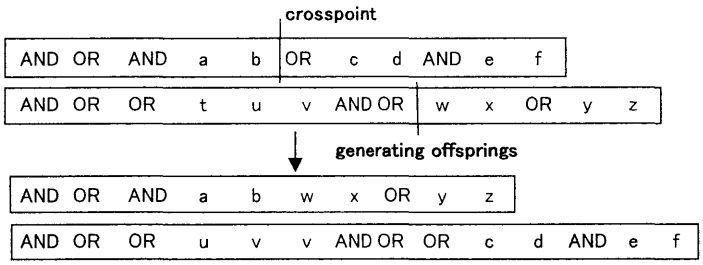
The mutation operation is used to reintroduce some diversity in an otherwise stagnant population. Then, we apply two kinds of mutation operations in the following.

(Global mutation : G-mutation)

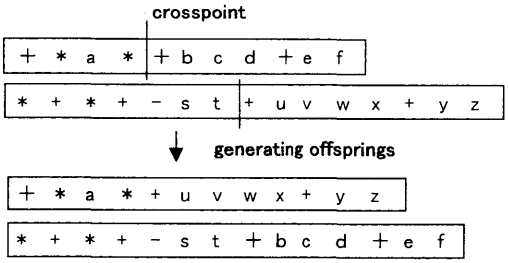
Generate a individual I_s, and select a subtree which satisfies the consistency of prefix representation. Then, select at random a terminal in the individual, and replace the terminal by the subtree of the individual I_s.

(Local mutation: L-mutation)

Select at random a locus in a parse tree to which the mutation is applied, we replace the place by another value (a primitive function or a variable). For example, we replace a logical operator in a individual in Level 1 by another logical operator. In a similar manner, a variable is replaced by another variable in a arithmetic expression. The selection of individual and the location of mutation operation are decided by using random numbers.



(a) GP for logical formula



(b) GP for arithmetic expression

Fig.3-Crossover operation.

3.2 Algorithm of GP

The crossover operation in the GP means exchanging of a part of tree in individual A by a part of individual B. These newly generated individuals replace old individuals with lower fitness.

Then, overall algorithm is summarized as follows. We iteratively perform the following steps until the termination criterion has been satisfied.

(Step 1)

Generate an initial population of random composition of possible functions and terminals for the problem at hand for the individuals in Level 1 and Level 2. The random tree must be syntactically correct program.

(Step 2)

Execute each individual (evaluation of production rules) in population and assign it a fitness value giving partial credit for getting close to the correct output. Then, sort the individuals according to the fitness S_i.

(Step 3)

Select a pair of individuals chosen with a probability p_i based on the fitness. The probability p_i is defined for ith individual as follows.

$$p_i=(S_i-S_{min})/\sum S_i(S_i-S_{min}) \tag{4}$$

where S_{min} is the minimum value of S_i, and N is the population size. Then, create new individuals (offsprings) from the selected pair by genetically recombining randomly chosen parts of two

existing individuals using the crossover operation applied at a randomly chosen crossover point. The algorithm is the same as the roulette strategy.

If the individual having highest fitness is not included, then we apply the strategy of elite preservation. Iterate the procedure several times to replace individuals with lower fitness.
(Step 5)

Then apply the GA to determine the relevant values for the constants included in the system equations. These parameters are represented as the numerical values in a string (individual).

4 Application to trading model induction

4.1 Wavelet transform of stock price

As the first application, we generate the prediction method for the stock price based on the if-the rules generated by the GP. The input data for the prediction is the wavelet transform of the stock price, and the output of the production rules is the trend of the stock price. Even though, we can prepare numerical method to predict the stock price, but here we have a kind of linguistic rules to explain the prediction of stock price [11].

Since the wavelet coefficients describe the short-term feature such as the short-term spectrum, it is relevant to use the wavelet coefficients to predict future stock prices. This pre-processing of the short-term feature of the stock price provides many advantages to the inference system over the prediction system which use only the observed stock prices based upon the adaptive method such as the neural networks.

In this paper we use the wavelet coefficients as the input variables to the multi-stage fuzzy inference system. The wavelet transform of the time series $x(t)$ is defined as follows [11].

$$x(t)=\sum_n\sum_m x_n^m\psi_n^m(t). \tag{11}$$

$$x_n^m=\int_{-\infty}^{\infty}x(t)\psi_n^m(t)dt. \tag{12}$$

$$\psi_n^m(t)=2^{m/2}\psi(2^m t-n). \tag{13}$$

where integer m and n are the dilation and translation index. For simplicity, we assume that the sampling interval of the time series $x(t)$ is 1. Since the Fouries Transform of each wavelet function ψ_n^m is not overlapped, then the integer m and n are restricted as follows.

$$m=2^K,\ \ n=Jm(J=1,\ 2,\ \ldots,\ K=0,\ 1,\ \ldots) \tag{14}$$

In the inference system, the wavelet coefficients obtained from the stock trends are used as the input variables. In a real application, the number of available coefficients is restricted. As is shown in equation (14), the maximum number of the dilation index m is defined by using the length L of the time series $x(t)$. Namely, the number of wavelet coefficient for the given dilation index m is determined by

$$N_m=L/2^m \tag{20}$$

If we choose N_m by considering the feature extraction in the time domain, then the maximum number of m is determined that corresponds to the resolution in the frequency domain.

Since we usually find structural changes of stock price in several years, the maximum value of L is limited to about to 1000. Then, we determine that the maximum number of m is 6.

4.2 Prediction of rise/fall of Price

The task of the simulation study is to evaluate the probability of correct prediction of the future stock price $S(t+T)$ where t and $t+T$ are the present time and the future.

The followings are problems to evaluate the inference system, besides the data set for learning and testing.

(1) definition of rise/fall.

(2) how far from the present time.

The first one is how to define the range of change of stock price (rise or fall). Then we introduce a threshold value $\theta(T)$ for the price change $S(t+T)-S(t)$. The threshold value may affect the performance of the inference. We define two threshold value for $\theta(T)$ independent of t.

In case of rise of stock price, we use $U(T)$ for $\theta(T)$. In case of fall of stock price, we use $D(T)$ for $\theta(T)$. For example, if the inference system estimate that $S(t+T)-S(t)>U(T)$ is held, then the system predict a rise of stock price. On the other hand if $S(t+T)-S(t)<D(T)$ is estimated, then the system predict a fall of stock price.

Related to the second problem, we choose several value of T to evaluate the inference. By taking several combination among $\theta(T)$ and T, we find a kind of dependency of inference on the value of T. For simplicity, we consider only the rise and fall of stock price. In the learning process, the prescribed value one (zero) is assigned for the case of rise and fall of stock price. Then, the parameters of the inference system are so adjusted that the output of the inference system predicts rise/fall of price, respectively by using the data for learning.

4.3 Result of Simulation Study

In the real application, the data for learning and the data for testing are exclusively selected from the database. But, in the simulation study, we show also the result of inference for the cases where the data for learning is also used for testing. Then, we distinguish following two cases of test.

(Test A) data for learning and is included also for testing

(Test B) data are exclusively selected for learning and testing

The data for learning and testing is selected as follows.

Number of stocks(chips): 100

Category of stock: stocks of electronics, machinery and trading companies

Duration of stock prices: a stock trend includes 1000 samples

Number of stock trend for a stock: two stock trends for a stock

Stock price is normalized: $0<x(t)<150$

The data for learning and testing is organized by using the published data on a CD-ROM data recorded from 1975 to 1986 [13]. We use the closing price of stocks in a trading day (daily price of stocks). Therefore, the unit of the time t and T are a day. In our analysis, the stock trend includes 1000 prices, then we have two stock trends for a stock recorded. These two stock trends are regarded as different trends. Then we select 100 stocks for which we apply the prediction from the CD-ROM data by deleting stock price with very little change during the period [12]. The category of industry whose stock price is used are electronics, machinery and trading companies.

We at first detect the time points where the rise/fall of stock price is observed. The number of data (time points of rise/fall) for learning and testing detected in a stock trend is not uniform. Its maximum number is 44, and the minimum number is 9. Its average is about 20. We call these points as the feature points. Since one stock trend has about 20 feature points and we have 100 stock trends, then the total number of feature points for learning and testing is about 20000. The feature detected in these feature points is transferred to the learning and testing system. The stock trend for learning and testing are selected at random from the pool of stock trends, and is used to evaluate the inference system. Namely, we select a stock trend for learning, then apply the inference system to another stock trend.

The parameters of simulation are selected as follows.

Maximum size of array (Level 1): 20

Maximum size of array (Level 2): 20

Population size in GP: 500

Probability of mutation: 0.01

Maximum generation of GP: 500

Table 1 and 2 show the ratio of correct prediction of stock price for Two-way prediction (Test A and Test B) depending on the combination of U, D and T. Table 3 shows the details of correct prediction for typical cases of rise and fall of stock prices. As is seen from the result, the stock price is predicted in average at the probability 70% (Test A) and 68%(Test B). We find no significant difference of prediction depending on the combination among $\theta(T)$ and T. The fact means if we choose $20\leq T\leq 50$, $15\leq U\leq 20$, and $10\leq D\leq 20$, we can predict future price change at the probability of about 70%.

Table 1. Two-way prediction, Test A(%)

T	U=20, D=10	U=15, D=15	U=20, D=20
20	74	73	75
30	71	67	71
50	70	66	65

Table 2. Two-way prediction, Test B (%)

T	U=20, D=10	U=15, D=15	U=20, D=20
20	71	65	65
30	67	64	63
50	64	62	62

5 Application to bond rating

5.1 Bond rating and financial ratio

The method is then applied to generate the if-then rules to classify the corporate bonds (rating). A rating simply helps investors determine the relative likelihood they might lose money on a given fixed income investment. In the financial market, the corporate bonds are evaluated and denoted by a kind of rating symbols ranging from AAA to CCC (AAA as the best company CCC as the worst one) to guarantee the profitability. Usually, the rating agencies have specialized sectors for rating which are processing a lot of checklists. Therefore, it is relevant to introduce an automatic system for rating by using published data.

The input data for the system is the financial indicators, and the output of the decision is the rating.

We must know the reasonable decision rules to explain the rating by using the financial ratios.

At first, we select three groups of Japanese companies[14] [15]. 1) 35 companies from electric machinery industry (group 1) 2)30 companies from mechanical machinery industry (group 2) 3)55 comapnies from several industries. (Group 3) 4) all 120 companies included in Group 1, 2 and 3 (group 4)

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.

If we select the companies from a single group, the rule prediction of rating is expected to be good, but it does not ensure the robustness of the prediction. Then, we select the companies partly from a single industry, and the rest of the companies are selected from a pool of several industries.

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.

In these process of preparing of data for learning, we at first select 19 kinds of financial ratio from the published CD-ROM data. By applying the statistical testing, we reject financial ratios having many outlyer. Finally, we select 7 financial ratio as the input to the inference system.

The information of rating for the companies are obtained from the published data of Japan Credit Rating Agency. 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 CCC or BBB. The Japanese companies treated here are relatively good, and the range of rating is included in AAA, AAA, A and BBB. The output is categorized into these three categories.

5.2 Result of inference

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).

We consider two cases of simulation study.

(Case 1) The data for learning (the number is N_1) and the data for testing (N_2) are exclusive, and these numbers are comparable.

(Case 2) About half of learning data is mixed into the testing data, and a part of the learning data is also used as the testing data.

All data are selected from Group 4. Table 3 show the result of simulation for Case 1. Table 4 show the result of simulation for Case 2. As is seen from the result, the correct recognition for rating is around 70%, and the result for Case 2 is slightly better than Case 1.

Table 3. Result of inference (Case 1)

N_1	N_2	Recognition
60	60	72

Table 4. Result of inference (Case 2)

N_1	N_2	Recognition
60	90	74

6 Discussion

In the following, our method is compared with conventional results. It should be considered that the rules treated here are linguistically expressed, and are not simply the numerical predictions. It is relevant that the of performance is compared with a similar inference systems.

Among linguistic inference systems, the fuzzy inference systems have been successfully applied to various fields. In previous works, we demonstrated that the multi-stage fuzzy inference systems are available for suppressing the number of rules by utilize input variables in a distributed manner [15] [16].

Therefore, the result of the paper is compared with the inference obtained by the multi-stage fuzzy inference systems. We suppose that the stages of the multi-stage inference system is three, and the input variables are allocated to each stage in a distributed manner. The number of membership function is assumed to be five. The weights included in the inference system are optimized by the backpropagation algorithm proposed in [15] [16]. An the same time the shape of the membership functions are optimized by the GA.

The simulation studies for the same example are summarized in Table 4 and 5 (Table 4 for the prediction of stock price, and Table for for bond rating).

As is seen from the results, the inference systems proposed in the paper have comparative performance to the conventional multi-stage fuzzy inference systems.

Table 5.Two-way prediction, Test A(%)

T	U=20, D=10	U=15, D=15	U=20, D=20
20	76	75	75
30	74	73	71
50	76	72	73

Table 6. Two-way prediction, Test B (%)

T	U=20, D=10	U=15, D=15	U=20, D=20
20	67	67	68
30	65	69	71
50	67	71	70

Table 7 and 8 show the result for rating by using the multi-stage fuzzy inference systems. The parameters for simulation study is the same as the system treated in the paper. As is seen by comparing the result in Table 3 and with Table 7 and 8, the performance of both inference systems are comparable.

Table 7. Result of inference (Case 1)

N_1	N_2	Recognition
60	60	76

Table 4. Result of inference (Case 2)

N_1	N_2	Recognition
60	90	78

7 Conclusion

This paper showed the GP procedure to generate the production rules and then apply the method to the prediction of stock price and rating analysis. The tree structure presenting the production rules are represented by a string, and the GP operations were applied to improve the fitness of the inference system. The results showed relatively good performance of the system.

For further works, the function set will be extended to slightly complicated functional forms rather than fundamental arithmetic.

References

[1] J.R.Quinlan: "Induction of Decision Tree", Machine Learning, 1, pp.81-106, 1986.

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

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

[4] M.Oussaidene, B.Chopard, O.V.Pictet and M.Tomassini: "Parallel genetic programming : An application to trading model evolution", Proc. of First Annual Conference of Genetic Programming, pp.357-362, 1996.

[5] M.Oussaidene, B.Chopard, O.V.Pictet and M.Tomassini: "Parallel genetic programming and its application to trading model induction", Parallel Computing, vol.23, pp.1183-1198, 1997.

[6] S.Tokinaga and S.Matsuno: "Generating the production rules by using the genetic programming and its application to risk analysis", Proc. of ITC-CSCC '01, vol.2, pp.961-964, 2001.

[7] J.R.Koza: *Genetic Programming*, MIT Press, 1992.

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

[9] Y. Ikeda and S.Tokinaga: "Approximation of chaotic dynamics by using smaller number of data based upon the genetic programming", Trans. IEICE, vol.E83-A, no.8, pp.1599-1607, 2000.

[10] Y. Ikeda and S.Tokinaga: "Controlling the chaotic dynamics by using approximated system equations obtained by the genetic programming", Trans. IEICE, vol.E84-A, no.9, pp.2118-2127, 2001.

[11] Y.Kishikawa and S.Tokinaga: "Prediction of stock trends by using the wavelet transform and the multi-stage fuzzy inference system optimized by the GA", Trans. IEICE, vol.E83-A, no.2, pp.357-366, 2000.

[12] Stock prices in CD-ROM, Toyokeizai Data Bank, 2000.

[13] Japan Credit Rating Agency: Annual Report, 2000.

[14] K.Tan and S.Tokinaga: "Optimization of fuzzy inference rules by using the genetic algorithm and its

application to the bond rating", J. of Operations Research Society of Japan, vol.42, no.3, pp.302-315, 1999.

[15] 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", Trans. IEICE, vol.E82-A, no.9, pp.1865-1873, 1999.

[16] Y.Kishikawa and S.Tokinaga: "Approximation of multi-dimensional chaotic dynamics by using the multi-stage fuzzy inference systems and the GA", Trans. IEICE, vol.E84-A, no.9, pp2128-2137, 2001.