Simple Learning Classifier Machine

Lubna.Z.Bashir* & Hind .A.Alrazzaq*

Received on: 8/3/2009
Accepted on: 16/2/2010

Abstract

A learning classifier system is one of the methods for applying a genetic-based approach to machine learning applications. An enhanced version of the system that employs the Bucket-brigade algorithm to reward individuals in a chain of co-operating rules is implemented and assigned the task of learning rules for classifying simple objects. The task is to classify an object that has one or more of the following features: wing, 2-legs/wheels, 3-legs/wheels, 4-legs/wheels, big, flies into one of the following: bird, vehicle. the main goal is to exploit the ability of the algorithm to perform well in a noisy environment and its ability to make little or no assumption about its problem domain. Results are presented which show that the system was able to learn rules for the task using only a few training examples and starting with classifiers that were randomly generated. It is argued that a classifier based learning method requires little training examples and that by its use of genetic algorithms to search for new plausible rules, the method should be able to cope with changing conditions. Results show also The parallel implementation of the algorithm would speed up the training process.

Keywords: Classifiers, Bucket brigade, Genetic Algorithms, Machine Learning

مكننة تصنيف تعليمية بسيطة

الخلاصة:

أنظمة التصنيف التعليمية أحدى الوسائل التي تستخدم الخوارزمية الجينية في تطبيقات تعليم المكننة.يعزز النظام بتوظيف خوارزمية أل (Bucket Brigade) في مكافأة القوانين التعليمية لتنفيذ مهمة القوانين التعليمية لتصنيف كائن بسيط.مهمة النظام تصنيف كائن يحمل مجموعة من الخصائص : له اجنحة ، له عجلتين ، ثلاث عجلات ، اربع عجلات ، كبير ، يطير و يصنفه الى طير او مركبة. الهدف الرئيسي هو اختبار قابلية الخوارزمية لكفاءة الاداء في بيئة مفعمة بالضجيج. النتائج بينت قابلية النظام في تعلم القوانين لأداء مهمته في تصنيف كائن بسيط باستخدام أمثلة تعليمية قليلة والبدء بأمثلة تصنيفية متولدة عشوائيا. طرق التعليم المعتمدة على المصنفات تتطلب بعض الأمثلة التعليمية وتستخدم الخوارزمية الجينية للبحث عن قوانين جديد أفضل تتلائم مع الشروط المتغيرة.النتائج بينت كذلك أن المعالجة المتوازية للخوارزمية تسرع من عملية التعليم.

Introduction

A Classifier Learning system is one of the ways of using evolutionary methodology for machine learning applications. These systems are a class of rule-based, message processing systems, Rules in these systems are known as classifiers because they are mainly used to classify messages into general sets. Many rules could be

* **Building and Construction Engineering Department, University of Technology / Baghdad**

https://doi.org/10.30684/etj.28.9.14
2412-0758/University of Technology-Iraq, Baghdad, Iraq

active simultaneously since the only action of an active classifier is to post a message onto a message list**[3]**.

Learning in classifier systems is achieved by two mechanisms: **Bucket-brigade** and **Genetic Algorithms, [5].** Bucket-brigade is designed for allocating credit to classifiers or rules according to their usefulness in attaining systems goals. It is also designed as a fitness function; rating each classifier in the system. Genetic Algorithms are used to search for new plausible classifiers or rules by recombining good classifiers to produce new ones**[9]**.

An enhanced version of the algorithm is implemented and assigned the task of evolving learning rules for classifying simple objects described in terms of their features. Results are presented that demonstrate the ability of the method to evolve learning rules for the given task. It is argued that the algorithm does not require a lot of training examples. However, the results indicate that the effectiveness of bucket-brigade as a fitness function and as a mechanism for rewarding rules in the type of application considered needs further investigation.

What is Learning Classifier Systems

A classifier system (sometimes referred to as ***adaptive agent***) is a specialized application of genetic algorithm. It is essentially a way to make genetic algorithms more adaptive to a dynamic environment. A classifier system has the ability to categorize its environment and create rules dynamically, thus making it able to adapt to differing circumstances **[7]**.

GAs have been used extensively in genetic based machine learning systems (often referred to as learning classifier systems or LCS). Learning classifier systems are simple event driven rule-based systems which use a GA to evolve their rules, the aim being to evolve better and better rules over time. The GA is applied at specific times to the rules (which become members of the current population). The new rule set is then generated using crossover and mutation operators. A classifier system also consists of a rule and message system, a reward mechanism (for successful rules) and some way of obtaining input and generating outputs. **[3,7]**.

Classifier Systems Components

Learning Classifier systems are a form of adaptive system introduced by John Holland in the mid1970s as a form of domain - independent learning system. A classifier system is used as a machine learning system that learns syntactically simple string rules (called classifier) to guide its performance in an arbitrary environment. A classifier system consists of three main components as illustrated in **Fig.1[11]:**

1. Performance system (Rule and message system).

2. Apportionment of credit system (Bucket Brigade algorithm).
3. A rule discovery (the genetic algorithm).

1. The Performance System

The performance system (also called rule and message system), is a special kind of production system. A production system is a computational scheme that uses rules as its only algorithmic device. The rules generally have the form:
If <condition> then <action>

The meaning of a production rule is that that action may be taken (the rule is fired) when the condition is satisfied **[3,7]**.
The performance system is composed of: ***classifier store, message list, detector, effector.***

- **Classifier Store**

The classifier store is the system's long term memory. It is made up of a population of classifiers. A classifier is made up of one or more conditions (known as the ***condition part***) and one action (called the ***action part***). The condition part specifies the set of messages to which a classifier is sensitive, and the action part indicates the message it will broadcast or send out when its condition part is satisfied. Thus a classifier list consists of one or more classifiers of the form:

$$C_1, C_2, \ldots\ldots\ldots .. C_n / a$$

Where:

$C_1, C_2, \ldots C_n \{n >= 1\}$ are the conditions making up the condition part and $'a'$ is the action part, conditions are connected by AND operator. The'/' indicates a separation between conditions and action. Each C_i is a string of fixed length K over a fixed alphabet. In most practical systems, the string is defined over three alphabets: {1,0, #}. The '#' is don't care (wild card) symbol that can match any of the chosen symbols {0 or 1}**[9]**.

For example:

A classifier having a condition represented by: **1#01**
Will recognize (match) the following messages: **1101** and **1001**
A classifier posts one or more messages onto the message list when it is activated.
The action part of a classifier is used to form the message it sends out when it is activated. It is also a string of fixed length K defined over the alphabet {1,0}**[9]**.

- **Message List**

The message list acts as the system's short-term memory and as the medium for communication between classifiers, and the output interface. It is made up of external messages (input observations) and internal messages (messages from classifiers). A message is represented by a string of fixed length K (same length as that for a condition) over the same set of alphabets {1,0} as the action**[9].**

- **Input Interface (Detector)**

  This receives the input messages from the environment and transforms them into fixed length strings to be placed on the message list**[9].**

- **Output Interface (Effector)**

  Messages placed on the message list by classifiers are processed through the output interface in order to communicate with the system's environment **[9]**.

The basic structure of the rule and message system (performance system) is presented in ***Fig 2***. To see how this rule and message system works assume a binary internal representation **[8]**:

The whole process is initiated by an event occurring and being detected by the "Detectors". These translate the event into an internal representation which will be used to determine which classifier (or rule) should be fired. Next each of the classifiers in the classifier list is compared with the translated event. Any classifiers whose conditions match the event may be fired. In the above example, the first classifiers' condition matches the event. This is because 10# matches any event in which the first two elements are 1 and 0. Thus it can match 101 and 100.

Once a classifier condition is matched, that classifier becomes a candidate to post its message to the message list at the next time step. Whether the classifier does post its message to the message list depends on the outcome of the activation action. The activation auction depends on the classifier value (or weighting). This value is based on the credit, which has been assigned to it. The approach used to determining the assignment of the credit to increase a classifiers value is the Michigan approach (or Buck Brigade)**[6]**.

2. ***Apportionment of Credit System (Bucket Brigade algorithm).***

The Bucket-brigade algorithm is designed to solve the credit assignment problem for classifier systems and to determine the worth of each classifier. The credit assignment problem is that of deciding which of a set of early active classifiers should receive credit for setting the stage for later successful actions. The BBA service economy contains the following components: ***An auction, a clearinghouse, taxation*** **[2].**

- **An Auction**

When condition parts of classifiers are matched by one or more messages they do not directly post their messages. Instead, having its tradition matched qualifies a classifier to participate in an activation auction in which it maintains a record of its net worth, called its strength. Each matched classifier makes a ***Bid*** proportional to its strength in this way; rules that are highly fit (have accumulated a

large net worth) are given preference over other rules. The auction permits appropriate classifiers to be selected to post their messages. A classifier's bid depends on its strength and the specificity of its condition. The specificity measures the relevance of a classifier's condition to a particular message. Formally, a classifier's bid is defined as a product of C_{bid} and a linear function of classifier's specificity, and classifier strength **[2].**

$$Bid_i = C_{bid} * f(SP) * S_i \quad \ldots\ldots(1)$$

$$f(SP) = bid_1 + bid_2 * SP \quad \ldots\ldots(2)$$

Where: C_{bid} is the bid coefficient, usually C_{bid} a constant less than 1. Specificity (SP) = number of non # /Total Length of condition part. bid_1, bid_2 Are input parameters. S Is strength, i is classifier index.

The effective bid (EB) for each matched classifier is the sum of its deterministic bid and a noise term:

$$EB_i = Bid_i + N(s_{bid}) \quad \ldots\ldots(3)$$

Where the noise N is a function of the specified bidding noise standard deviation s_{bid} .

The winning classifiers place their messages on the message list and their strengths are reduced by the amount of their bids. Typically, if $S_C(t)$ denotes the strength of a winning classifier, C , at time t and $B_C(t)$ denotes its bid at the same time t , then its strength at time $t+1$ is:

$$S_C(t+1) = S_C(t) - B_C(t) \quad \ldots\ldots(4)$$

In other words a winner pays for posting a message on the message list **[2,9]**.

- **Clearinghouse**

The selected classifier must clear its payment through the clearinghouse, paying its bid to other classifiers for matching message rendered. A matched and activated classifier sends its bid to those classifiers responsible for sending the messages that matched the bidding classifier's condition. The bid payment is divided in some manner among the matching classifiers. This division of payoff among contributing classifiers helps ensure the formation of an appropriately sized sub population. Thus different types of rules can cover different types of behavioral requirements without undue inter species competition **[9].**

For example, if C' sent the message matched by C above, then the strength of C' at time $t+1$ is:

$$S_{C'}(t+1) = S_{C'}(t) + B_C(t) \quad \ldots\ldots(5)$$

When system goals are attained, a pay-off (reward or punishment) is added to the strength of all the classifiers that are active at that time **[9]**.

•Taxation

Each classifier is taxed to prevent freeloading, thereby biasing the population toward productive rules. Many schemes are available, a tax was simply collected proportional to the classifier strength:

$$T_i = C_{tax} * S_i \quad \text{.....(6)}$$

Where: T is tax, C_{tax} is coefficient of tax, S is strength, i classifier index.

There are two types of tax:

A. ***Life Tax*** removes redundant classifiers in the system and encourages the more frequent acting classifiers .It removes a tax at each time step from every classifier in the population, so fervors classifiers that can replace the lost strength. Unfortunately, it weakens classifiers in infrequent niches and chains starting classifiers. Life tax controls classifiers that ***never*** bid whereas, bid tax control classifiers that bid too much.

B. ***Bid Tax*** is removed from classifiers in the match set that bid to control the auction. It is separate from the bid, which is only removed from ***unsuccessful*** active classifiers. The problem of over generals can be reduced as their strength is reduced to a greater extent than accurate generals (including those in default hierarchies). The bid tax sets a balance between specific and defaults classifiers' stable strength and so controls the likelihood of selection in rule discovery. This balance can be poor, as it is set prior to training **[3].**

To implement a well-defined procedure the auction and payment scheme was detailed. Classifiers make ***bids*** (B_i) during the auction. Winning classifiers turn over their bids to the clearinghouse as ***payments*** (P_i). A classifier may also have ***receipts*** (R_i) from its previous message- sending activity or from environmental reward. In addition to bids and receipts, a classifier may be subject to one or more ***taxes*** (T_i) taken together, the equation governing the depletion or accretion of the ith classifier strength as follows **[3]:**

$$S_i(t+1) = S_i(t) - P_i(t) - T_i(t) + R_i(t) \quad \text{.....(7)}$$

The apportionment of credit equation recasts into a more useful form where all payments and taxes have been replaced by their strength equivalent **[1]**.

$$S(t+1) = S(t) - C_{bid} * S(t) - C_{tax} * S(t) + R(t) \quad \text{.....(8)}$$

3.A Rule Discovery (the Genetic Algorithm).

- ***Selection***

The purpose of selection is, of course, to emphasize the fitter individuals in the population in hopes that there offspring will in turn have even higher fitness: There are several schemes used in GA: Roulette wheel selection, Sigma scaling techniques, Tournament selection, Ranking methods, Elitism, Boltzmann selection, Steady state selection.

Holland's original GA used fitness- proportionate selection, in which the expected value of an individual (i.e. , the expected number of times an individual will be selected to reproduce) is that individuals fitness is divided by the average fitness of population. The most common method to implement this is '***roulette wheel"*** sampling **[11]**:

Roulette Wheel Selection:

Each individual is assigned a slice of a circular "roulette wheel" the size of the slice being proportional to the individual's fitness. The wheel is spun N times, where N is the number of individuals in the population. On each spin, the individual under the wheel's marker is selected to be in the pool of parents for the next generation **[1]**.

- ***Crossover***

in contrast is applied with high probability. It is a randomized yet structured operator that allows information exchange between points. Simple crossover is implemented by choosing a random point in the selected pair of strings and exchanging the sub strings defined by that point. ***Fig. 3*** shows how Crossover mixes information from two parent strings, producing offspring made up of parts from both parents. This operator which does no table lookups or backtracking, is very efficient because of its simplicity**[3].**

- ***Mutation***

as in natural systems, is a very low probability operator and just flips a specific bit. Where the bits at one or more randomly selected positions of the chromosomes are altered. The mutation process helps to overcome trapping at local maxima. ***Fig. 4*** shows the effect of the one-bit mutation operator in the binary case **[3]**.

Basic Execution Cycle of a Learning Classifier System

The basic execution cycle of a learning classifiers system is:

(i) Place message(s) from input interface onto message list.
(ii) Compare message(s) on the message list with conditions of all classifiers and record matches.
(iii) Make classifiers bid for the right to recode messages.
(iv) Generate new messages.
(v) Adjust strengths of classifiers using the bucket-brigade algorithm.
(vi) Replace old message list by the new message list.
(vii) Process message list through the output interface.
(viii) If no classifier fired, use genetic algorithm to generate one or more new classifiers to replace the weakest classifier or classifiers.
(ix) Return to step (i) or stop if no more messages.**[4, 9].**

Classification Simple Objects (CSO)

The learning classifier system is called classification simple objects (CSO) assigned the task of inducing classification rules for classifying simple objects described in terms of 6 attributes. The task is to classify an object that has one or more of the following features: wing, 2-legs/wheels,3-legs/wheels, 4-legs/wheels, big, flies into one of the following: bird, and vehicle.

Coding (LCS – CSO) Conditions

The learning classifier system (LCS - CSO) receives 6 – bit messages from environment mapping it to 64 states from 0 to 63of six bits only. The six bit represents as fallows:

- First bit represented is object has wing (1 means the object has wing, 0 means that object has not wing).
- Second bit represented is object has 2 legs wheel (1 means the object has 2 legs wheel, 0 means that object has not 2 legs wheel).
- Third bit represent is object has 3 legs wheel (1 means the object has 3 legs wheel, 0 means that object has not 3 legs wheel).
- Four bit represent is object has 4 legs wheel (1 means the object has 4 legs wheel, 0 means that object has not 4 legs wheel).
- Five bit represent is object big (1 means object is big, 0 means object is small).
- Six-bit represent is object flies (1 means object is flies, 0 means object is not flies).

Coding (LCS – CSO) Actions

LCS – CSO has an action consisting of only one bit, LCS – CSO action has the form and meaning as fallows:

- 1 means the object is bird.
- 0 means the object is vehicle.

Representation of (LCS – CSO)

Performance system of the ***LCS-CSO*** consists of a message list and classifier store. The classifier

list of LCS-CSO contain a set of rules called classifiers, which represents the knowledge and controller of the system at execution time. Condition part of classifier consists of **(6 bit),** and action part consists of **(1 bit)**. The size of classifier store for ***LCS-CSO*** will be **(64)** Rules and all classifiers have the same strength value at the beginning.

Example

The representation of the rule "If object wing AND small AND has not 2 leg wheel AND has not 3 leg wheel AND has not 4 leg wheel AND flies THEN the object is bird ":

Wing big 2wheel 3wheel 4wheel flies / bird

1 0 0 0 0 1 / 1

The Implemented of The (LCS – CSO) System

The characteristics of the enhanced version of the classifier learning system developed and implemented for the LCS – CSO are given below.

Classifier List of (LCS – CSO)

For the sake of simplicity, each classifier had one condition and one action. The condition was represented as a fixed string of characters defined over the alphabets {1, 0, .}. A '1' indicates a feature that must be present, '0' represents a feature that must be absent and '.' indicates a feature that may or may not be present. The action part was also represented as a string of characters over the same three alphabets as the condition part.

Message List of (LCS – CSO)

The message list was made up of training examples and messages generated by the classifiers. The training examples were not removed from the message list until the end of the learning session. They were 'fed' into the system again after each generation. This departs from the standard practice of generating a completely new message list at each generation. The procedure implemented here ensured that newly produced classifiers had a chance of bidding for and classifying the training examples no matter when they were generated, and it also ensured that messages which were not recognized at previous generations were not lost.

Initial Classifier and Classifier Strength (LCS – CSO)

An example pattern for each class was randomly chosen, and used to form part of the initial classifiers. Some features were randomly converted to don't cares and pass through. Pairs were then randomly selected from these initial classifiers and 'mated' to produce the remaining members of a fixed-size population of classifiers. Each

newly generated classifier is assigned a value which is modified by the bucket-brigade to reflect is usefulness. This value is the same initially for each classifier.

Bucket Brigade of (LCS – CSO)

The bucket-brigade algorithm was modified so that external payoffs (rewards or punishment), when received, are paid to the classifier responsible for the action that earned the payoff and to all those that had taken part (in previous generations) in helping to arrive at that goal.

It should be noted that in the standard classifier system, payoffs are added to the strengths of classifiers active at the time the payoff is received. The new measure was introduced to overcome some of the problems encountered during the initial stages of the experiments. It was discovered that some bad classifiers were rewarded simply because they happened to be active when payoff was received. It was designed to prevent this from happening. The measure was also designed to speed up the passage of reward or punishment to classifiers responsible for previous actions that led to the receipt of the payoff. In a standard implementation of the bucket-brigade algorithm, a particular sequence of actions will have to be repeated several times before payoffs could flow through to the classifiers that were active at the early stages.

Replacement of Classifiers of (LCS – CSO)

In addition to the strength of a classifier which is modified by the bucket-brigade, another metric called **no of wins** was introduced to determine when a classifier is to be removed from the fixed-size classifier list so as to have room for new ones. The metric is a simple count of the number of times a classifier succeeded in bidding for the right to place messages onto the message list. Whenever new classifiers are to be introduced into the system, those whose strengths fell below the fixed value assigned to each newly generated classifier and those whose no-of-wins is zero are replaced.

Standard classifier systems typically employ only the strength of a classifier to determine the 'life' span of the particular classifier. One major problem with this arrangement is that a classifier that is necessary for some sequence of actions (i.e. it is part of a chain) could be removed because its strength happens to be lower than other members of the population. The new metric is designed to overcome this problem.

Experimental Results

The whole project execute using Pascal language, Executing the (CSO) code, The system was able to

learn rules for the given task using only a few training examples and starting with classifiers that were randomly generated.

The system responds by presenting the initial report for LCS-CSO. The classifier system run for 1000 iterations, termination with the snapshot report for LCS-CSO ,the initial and last report display in ***Appendix .A***. The correct rules have achieved high strength values, by contrast the bad rules have strength and bid values near zero. The classifier system eliminates the bad rule quickly thereby achieving near perfect performance. the performance of the system after 1000 iteration illustrated in ***Table.1*** and data chart illustrated in ***Fig.5***.

Possible solutions to wrong conditions problem include:

- Careful selection of training set.
- Increasing the size of the training set.
- Controlling the application of genetic operators so that valid offspring are always produced.

Conclusion and Discussion

- A genetic based approach – an enhanced version of the Holland classifier learning system - was able to evolve learning classification rules.

- The use of Genetic Algorithms to search for new plausible rules, the method should be able to cope with changing conditions.

- The system was able to learn rules for the given task using only a few training examples and starting with classifiers that were randomly generated.

- The parallel implementation of the algorithm would speed up the training process.

Appendix .A

==========

(LCS-CSO) for Classifying Simple Object

==========

population parameters

---

number of classifiers = 37
number of positions = 6
number of action = 1
bid coefficient = 0.1000
bid spread = 0.0750
bidding tax = 0.0100
existence tax = 0.0200
generality probability = 0.5000
bid specificity base = 0.2500
bid specificity mult. = 0.1250
edid specificity base = 0.2500
ebid specificity mult. = 0.1250

environmental parameters

---

total number of signal = 6

apportionment of credit parameters

---
bucket brigade flag = false

reinforcement parameters
---
reinforcement reward = 10.0

Timekeeper parameters

Initial iteration = 0
Initial block = 0
Report period = 50
Console report period = 50
Plot report period = 50
Genetic algorithm period = 50

Genetic Algorithm Parameters

Proportion to select/gen = 0.4000
Number to select = 7
Mutation probability = 0.0200
Crossover probability = 1.0000
Crowding factor = 3
Crowding subpopulation = 3

snapshot report
[block: iteration] - [0:0]
current status
signal = 000000
Decoded signal = 0
desired output = 0
classifier output = 0
enviromental message: 000000

no.	strength	bid	ebid	M	classifier
1	10.00	0.00	0.00		000001:[1]
2	10.00	0.00	0.00		000011:[1]
3	10.00	0.00	0.00		000100:[0]
4	10.00	0.00	0.00		000110:[0]
5	10.00	0.00	0.00		001000:[0]
6	10.00	0.00	0.00		001010:[0]
7	10.00	0.00	0.00		001100:[0]
8	10.00	0.00	0.00		001110:[0]
9	10.00	0.00	0.00		010000:[0]
10	10.00	0.00	0.00		010010:[0]
11	10.00	0.00	0.00		010100:[0]
12	10.00	0.00	0.00		010110:[0]
13	10.00	0.00	0.00		011000:[0]
14	10.00	0.00	0.00		011010:[0]
15	10.00	0.00	0.00		011100:[0]
16	10.00	0.00	0.00		010110:[0]
17	10.00	0.00	0.00		011000:[0]
18	10.00	0.00	0.00		011010:[0]
19	10.00	0.00	0.00		011100:[0]
20	10.00	0.00	0.00		011110:[0]
21	10.00	0.00	0.00		100000:[1]
22	10.00	0.00	0.00		100001:[1]
23	10.00	0.00	0.00		100010:[1]
24	10.00	0.00	0.00		100011:[1]
25	10.00	0.00	0.00		1#####:[1]
26	10.00	0.00	0.00		#####1:[1]
27	10.00	0.00	0.00		1####1:[1]
28	10.00	0.00	0.00		####01:[1]
29	10.00	0.00	0.00		1###0#:[1]
30	10.00	0.00	0.00		#1####:[0]
31	10.00	0.00	0.00		##1###:[0]
32	10.00	0.00	0.00		###1##:[0]
33	10.00	0.00	0.00		#11###:[0]
34	10.00	0.00	0.00		##11##:[0]

35 10.00 0.00 0.00 #111##:[0]
36 10.00 0.00 0.00 ######:[0]
37 10.00 0.00 0.00 ######:[1]
new winner[1] : old winner[1]

Initial Report for LCS – CSO

[block: iteration] - [0:1000]
current status
---
signal = 011111
Decoded signal = 31
desired output = 0
classifier output = 0
environmental message: 011111

no.	strength	bid	ebid	M	classifier
1	30.99	0.00	0.00		0011#0:[0]
2	30.99	0.00	0.00		0101#1:[0]
3	0.00	0.00	-0.04	x	01#1#1:[0]
4	0.69	0.06	0.10	x	0111#1:[0]
5	18.56	1.67	1.54	x	0111#1:[0]
6	2.37	0.21	0.43	x	0111#1:[0]
7	0.00	0.00	0.00		#101##:[0]
8	18.56	1.91	1.95	x	011111:[0]
9	0.00	0.00	0.00		100000:[1]
10	0.90	0.00	0.00		11#1#1:[0]
11	2.37	0.21	0.11	x	0111#1:[0]
12	0.28	0.00	0.00		11#1#1:[0]
13	30.99	0.00	0.00		0111#0:[0]
14	18.56	1.67	1.62	x	0111#1:[0]
15	0.50	0.03	0.05	x	0#11##:[0]
16	2.37	0.21	0.12	x	0111#1:[0]
17	0.00	0.00	0.10	x	01#1#1:[0]
18	0.60	0.05	-0.01	x	0#11#1:[0]
19	30.99	0.00	0.00		0101#1:[0]
20	18.89	1.95	1.95	x	011111:[0]
21	0.19	0.02	0.01	x	0111#1:[1]
22	0.40	0.03	0.14	x	0#11##:[0]
23	0.31	0.00	0.00		11#1#1:[0]
24	0.35	0.03	0.12	x	0#11#1:[0]
25	0.28	0.03	-0.04	x	0111#1:[0]
26	1.66	0.00	0.00		0111#0:[0]
27	0.50	0.04	0.16	x	0#11#1:[0]
28	0.40	0.03	-0.09	x	0#11#1:[0]
29	30.99	0.00	0.00		0111#0:[0]
30	18.56	1.43	1.38	x	0111##:[0]

31 76.93 7.69 7.76 x 011111:[0]
32 0.50 0.04 0.07 x 0#11#1:[0]
33 18.56 1.67 1.78 x 0111#1:[0]
34 18.56 1.67 1.70 x 0111#1:[0]
35 30.99 0.00 0.00 0111#0:[0]
36 0.50 0.04 -0.04 x 0#11#1:[0]
37 30.99 0.00 0.00 0101#1:[0]

new winner[31] : old winner[31]

Last Report for LCS – CSO

References

[1][Bull.Larry,2004], "Learning Classifier Systems: A Brief Introduction", Faculty of Computing, Engineering & Mathematical Sciences University of the West of England, Bristol BS16 1QY, U.K. Larry.

[2][Bhatia. Nikhil, Dixit. Prashant, Sood . Mohit.1999]"GAMBLE: Genetic Algorithm Based Machine learning Expert" Indian Institute of Technology.

[3][Goldberg , David E. ,1989], "Genetic Algorithms in Search, Optimization & Machine Learning", Addison-Wesley Publishing Company, Inc.

[4][Holland, John H. 1986], "Escaping Brittleness: The possibilities of General - purpose Learning Algorithms Applied to Parallel Rule - based Systems", Machine Learning an Artificial Intelligence Approach Vol. II, pp. 593 - 623, ed. R.S. Michalski, J. G. Carbonnell and T. M. Mitchell, Tioga, Palo Alto, Calf.

[5][Holland, John H.; Holyoak, Keith J.; Nisbett, Richard E.; and Thagard, Paul R., 1986], "Induction Processes of Inference, Learning and Discovery", MIT Press, Cambridge.

[6][HuntJohn,2002],"learningclassifiers ystems"JaydeeTechnologyLtd,Hartham Park,Corsham,Wiltshire,SN13ORP.

[7][Jakobsen. Troels,2004],"Classifier System Abstracts "Aarhus school of business ,Denmark.

[8][Nagasaka. Ichiro & Kikuchi. Makoto & Kitamura. Shinzo,2002]"A Formal Analysis of Classifier System and Interface Between Learning System and Environment". Department of Computer and Systems Engineering, Kobe University.

[9][Odetayo, Michael O.1990], "On Genetic Algorithms in Machine Learning And Optimisation", PhD Thesis, University of Strathclyde, Glasgow, U.K.

[10][Riolo, Rick, L. ,1986], " A Package of Domain Independent Subroutines for Implementing Classifier Systems in Arbitrary, User-defined Environments", Logic Computers Group, Division of Computer Science and Engineering, University of Michigan

[11][Zhou. Qing Qing and Purvis. Martin,2004] "A Market-Based Rule Learning System" aGuangDong Data Communication Bureau China Telecom 1 Dongyuanheng Rd., Yuexiunan,

Guangzhou 510110, China, Department of Information Science, University of Otago, PO Box 56, Dunedin, New Zealand and/or improving the comprehensibility of the rules.

Table (1) the CSO system Performance after 1000 iterations

Number of iteration	Proportion correct
50	0.92000
100	0.96000
150	0.97333
200	0.98000
250	0.98400
300	0.98667
350	0.98857
400	0.99000
450	0.99111
500	0.99200
550	0.99273
600	0.99333
650	0.99385
700	0.99429
750	0.99467
800	0.99500
850	0.99529
900	0.99556
950	0.99579
1000	0.99600

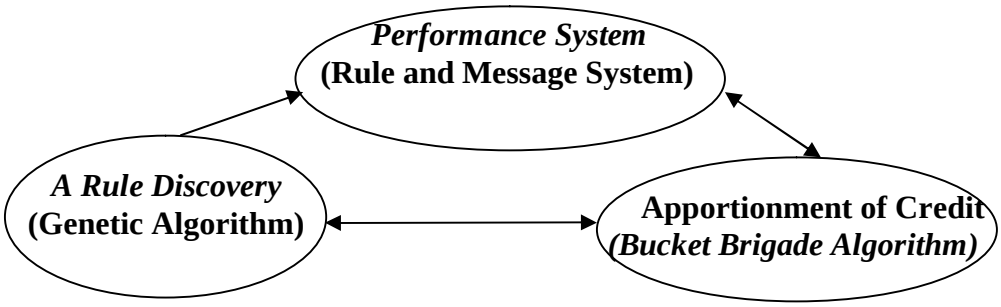

Figure (1) The Learning Classifier System.

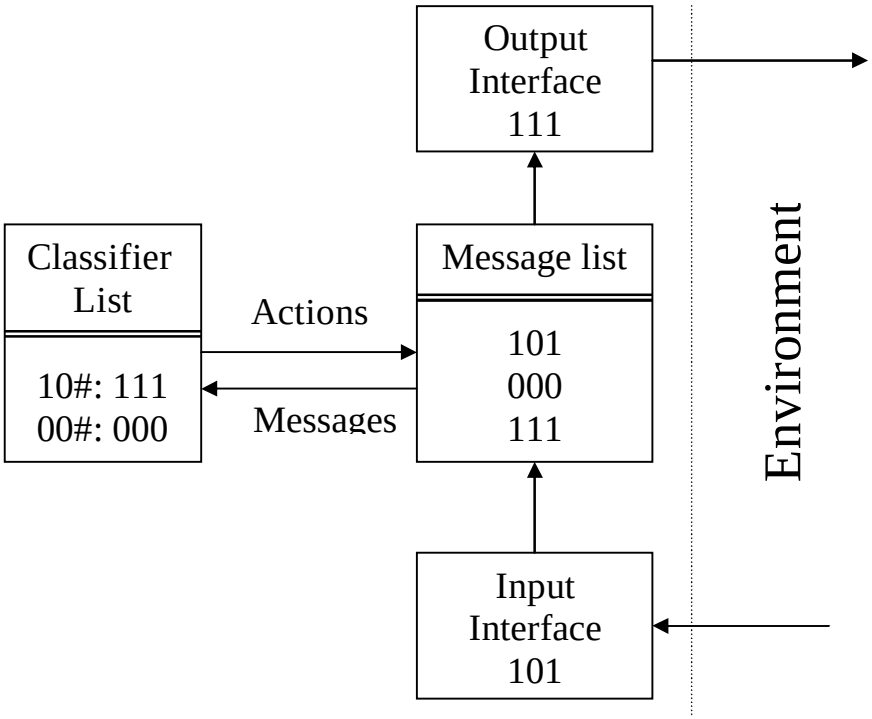

Figure (2) The performance system.

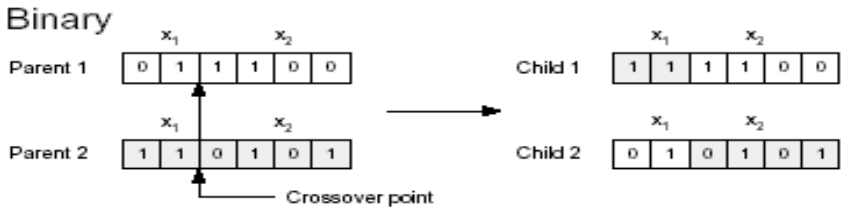

Figure (3) Crossover operator.

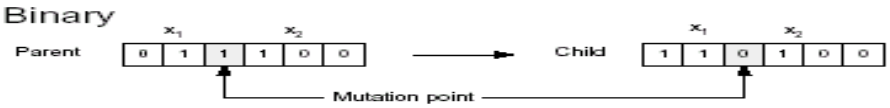

Figure (4) Mutation operator

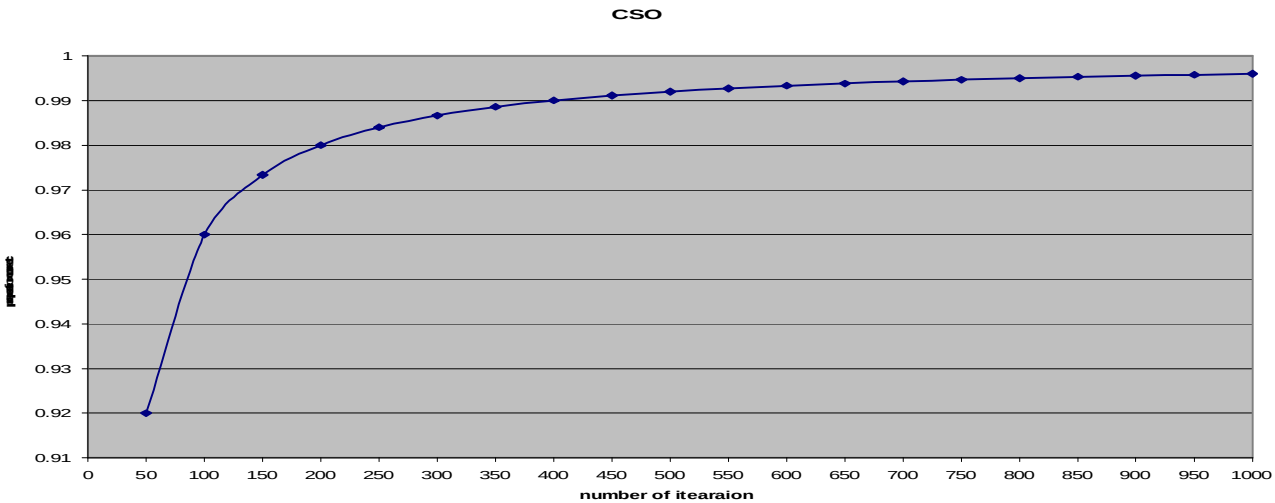

Figure (5) the CSO system Performance after 1000 iterations