World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:8, 2008

Learning Classifier Systems Approach for Automated Discovery of Censored Production Rules

Suraiya Jabin, and Kamal K. Bharadwaj

***Abstract*—**In the recent past Learning Classifier Systems have been successfully used for data mining. Learning Classifier System (LCS) is basically a machine learning technique which combines evolutionary computing, reinforcement learning, supervised or unsupervised learning and heuristics to produce adaptive systems. A LCS learns by interacting with an environment from which it receives feedback in the form of numerical reward. Learning is achieved by trying to maximize the amount of reward received. All LCSs models more or less, comprise four main components; a finite population of condition–action rules, called classifiers; the performance component, which governs the interaction with the environment; the credit assignment component, which distributes the reward received from the environment to the classifiers accountable for the rewards obtained; the discovery component, which is responsible for discovering better rules and improving existing ones through a genetic algorithm. The concatenate of the production rules in the LCS form the genotype, and therefore the GA should operate on a population of classifier systems. This approach is known as the 'Pittsburgh' Classifier Systems. Other LCS that perform their GA at the rule level within a population are known as 'Mitchigan' Classifier Systems. The most predominant representation of the discovered knowledge is the standard production rules (PRs) in the form of **IF P THEN D**. The PRs, however, are unable to handle exceptions and do not exhibit variable precision. The Censored Production Rules (CPRs), an extension of PRs, were proposed by Michalski and Winston that exhibit variable precision and supports an efficient mechanism for handling exceptions. A CPR is an augmented production rule of the form: **IF P THEN D UNLESS C**, where Censor C is an exception to the rule. Such rules are employed in situations, in which conditional statement **IF** P **THEN** D holds frequently and the assertion C holds rarely. By using a rule of this type we are free to ignore the exception conditions, when the resources needed to establish its presence are tight or there is simply no information available as to whether it holds or not. Thus, the **IF** P **THEN** D part of CPR expresses important information, while the **UNLESS** C part acts only as a switch and changes the polarity of D to ~D. In this paper Pittsburgh style LCSs approach is used for automated discovery of CPRs. An appropriate encoding scheme is suggested to represent a chromosome consisting of fixed size set of CPRs. Suitable genetic operators are designed for the set of CPRs and individual CPRs and also appropriate fitness function is proposed that incorporates basic constraints on CPR. Experimental results are presented to demonstrate the performance of the proposed learning classifier system.

***Keywords*—**Censored Production Rule, Data Mining, Genetic Algorithm, Learning Classifier System, Machine Learning, Pittsburg Approach, , Reinforcement learning.

Suraiya Jabin is a Ph.D. scholar and lecturer at Department of Computer Science, FMSIT, Jamia Hamdard, New Delhi, India (phone: +919810822834; fax: +91116089686 email: suraiya224@gmail.com).

Kamal K. Bharadwaj, is a professor at the School of Computer and Systems Sciences (SC & SS), Jawaharlal Nehru University (JNU), New Delhi, India (e-mail: kbharadwaj@gmail.com).

I. Introduction

DATA mining deals with the discovery of hidden knowledge, unexpected patterns and new rules from large databases [3]. The idea is to build computer programs that sift through databases automatically, seeking regularities or patterns [1].

Data Mining is regarded as the key element of a much more elaborate process called **Knowledge Discovery in Databases (KDD)** [3] which *is defined as the non – trivial process of identifying valid, novel, and ultimately understandable patterns in data*" by Fayyad (1996) [2]. Machine learning provides the technical basis of data mining. There is no magic in machine learning, no hidden power, and no alchemy. Instead, there is an identifiable body of simple and practical techniques that can often extract useful information from raw data. Some well known techniques are Decision trees, Classification rules, neural networks, statistical modeling, linear models, Instance based learning, Clustering, and Evolutionary Computation such as Genetic algorithms (GA) and Learning Classifier Systems (LCS) etc [1].

Learning Classifier Systems (LCS) [Holland, 1976] are a machine learning technique which combines evolutionary computing, reinforcement learning, supervised learning or unsupervised learning, and heuristics to produce adaptive systems. They are rule based systems, where the rules are usually in the traditional production system form of "IF premise THEN decision". An evolutionary algorithm and heuristics are used to search the space of possible rules, whilst a credit assignment algorithm is used to assign utility to existing rules, thereby guiding the search for better rules. The LCS formalism was introduced by John Holland [5] and based around his better known invention – the Genetic Algorithm (GA) [4].

In 1960's, John Holland, also known as father of Genetic Algorithm, introduced GA as method of studying *natural adaptive system* and designing *artifical adaptive systems*

based on *Darwinian natural selection* and *Mendelian genetics.* This method eliminates weak elements by favoring retention of optimal and near optimal individuals (survival of the fittest), and recombines features of good individuals to perhaps make better individuals. Genetic algorithms (GAs) use this method to search the representation space of artificial adaptitve system, That represent a problem's search space as sequences (strings) of symbols chosen from some alphabet (usually a binary alphabet). The algorithm performs optimization by manipulating a finite population of chrosomes. In each of a number of cycles called *generations*, the GA creates a set of new chromosomes by crossover, inversion and mutation, which correlate to processes in natural reproduction. In technology and science GAs have been used as adaptive algorithms for solving practical problems and also as computational models of natural evolutionary systems [10].

The application of genetic algorithms to machine learning has been addressed from two different approaches: the **'Michigan'** approach and the ***'Pittsburgh'*** approach.

The Michigan approach also known as classifier systems (CS) is based on the model introduced by Holland [4, 8]. A classifier system is an adaptive system that learns a set of rules by the interaction of the environment, from which it receives reward. Two main components can be distinguished:

- the *Credit Assignment Algorithm* which evaluates the efficiency of rules.
- the *Discovery component* performed by a genetic algorithm which acts on the rule level, seeking for new promising rules that can improve the system performance.

In the Pittsburgh approach also known as learning systems (LS), each individual represent a whole rule set instead of a single rule. This allows evaluation of an individual as a complete solution to the learning problem, avoiding thus the use of credit assignment algorithm. The fitness of each individual is evaluated as the performance of its rule set as a whole, considering different aspects as the classification accuracy, the no. of required rules etc. Both approaches have successfully been applied on a variety of classification problems [6]. Successful data mining applications of learning classifier systems have been shown in the past (Bernado, Llora, & Garrell, 2001) investigating and comparing performance of the accuracy-based ***Michigan-style LCS*** **XCS** (Wilson, 1995) and the ***Pittsburgh-style LCS*** GALE (Llora & Garrell, 2001). Both systems showed competent performance in comparison to six other machine learning systems [7].

The most predominant representation of the discovered knowledge is the standard production rules (PRs) in the form **If P then D**. The PRs, however, are unable to handle exceptions and do not exhibit variable precision [21]. Exceptions, which focus on a very small portion of a dataset, have been ignored or discarded as noise in machine learning, but the goal of KDD is broader and it is always interesting to discover exceptions, as they challenge the existing knowledge and often led to the growth of knowledge in new directions [22].

As an extension of PR, Michalski and Winston [11] have suggested Censored Production Rule (CPR) as an underlying representational and computational mechanism to enable logic based systems to exhibit variable precision, in which certainty varies, while specificity stays constant. A CPR has the form '**If P Then D Unless C'**, where C (censor) is the exception condition. Such rules are employed in situations, in which the conditional statement **'If P then D'** holds frequently and the assertion **C** holds rarely. By using a rule of this type we are free to ignore the censor (exception) conditions, when the resources needed to establish its presence are tight or there is simply no information available as to whether it holds or does not hold. As time permits, the censor condition **C** is evaluated establishing the conclusion D with higher certainty, if C does not or simply changing the polarity of D to ~D if **C** holds. For Example, consider the assertion that birds fly:

$\forall x$: is_bird (x) $\rightarrow$ flies(x)

This general assertion enables us to expect that any newly observed bird flies. But not all birds fly. For example penguins, ostriches, emus, kiwis and domestic turkeys do not fly. Even a flying bird cannot fly when it is dead, or sick or has broken wings. Now we can write:

$\forall x$: is_bird(x)$\rightarrow$ flies(x) $\llcorner$ is_ostrich(x)
V is_penguin(x)
V is_kiwi(x)
V has_broken_wings(x)
V is_dead(x)
V is_sick(x)

A CPR may have more than one censor conditions, say C_1, C_2, …..C_n and is denoted as:

If P **Then** D **Unless** C_1 V C_2 V……..C_n.

In this paper we have presented Pittsburgh style LCS for automated discovery of augmented production rules with exceptions in the form of CPRs.

II. LCS FOR CPR DISCOVERY

So many systems following LCS approach have been developed; one such system is EpiCS developed by John Homes, which was based on early works of Wilson. EpiCS was designed to use standard production and association rule as underlying knowledge but in our proposed work, designed automated system is using Censored Production Rules as underlying knowledge representation.

John Holmes developed a stimulus-response learning classifier system, BOOLE++ [15, 16] and later, EpiCS [19] which was based on Wilson's early work on the Animat [17], BOOLE [18] and NEWBOOLE [14]. Since the designed automated system is completely based on EpiCS, so some more light is thrown on EpiCS. Like other LCS, EpiCS's representation scheme expresses a rule as a condition-action pair, or a *classifier*, where in attributes are encoded as "genes". The left-hand side (condition) of the classifier is commonly referred to as a *taxon*, and the right-hand, the *action*. More commonly, the action is described as the *action*

bit, since the type of problems for which one uses EpiCS typically have a single, dichotomous classification, such as dead/alive, diseased/healthy, etc. Classifiers are contained in a *population* of constant size. EpiCS is constructed using the three-part framework of a typical LCS, the performance, reinforcement, and discovery components as shown in Fig. 1.

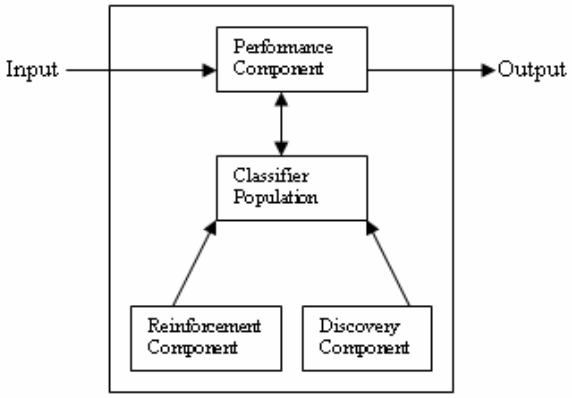

Fig. 1 High – level schematic of EpiCS [13]

A. Performance Component

The *performance component* creates a subset of all classifiers in the population whose taxa (premise part) match a stream of data received as input from the environment. In this way, the performance component is analogous to a *forward chaining rule-base* system. All classifiers in the population whose taxa match the input stream comprise a *Match Set* [M], even though some of these classifiers may advocate different actions. The process is equivalent to the triggering of rules, and [M] is analogous to an agenda in an expert system. From [M], the classifier with the proportionally highest strength is selected. The action of this classifier is then used as the output of the system; this process is analogous to the firing of a rule in an expert system [13].

B. Implementation

In classification problems, the environment is a *training set* of preclassified examples; each example is described by a vector of attributes and a *class* label; the goal of learning is to form a description that can be used to classify previously unseen examples with high accuracy [9]. Input to this automated censored production rule discovery system is a training set of preclassified examples. As an example consider the following bird training data set shown in Table I.

Chromosome representation Scheme: Since we have followed *Pittsburgh approach*, individual chromosome is represented by sets of CPRs as and a gene as one censored production rule.

A CPR or gene is divided into three parts: IF part, CENSOR part, THEN part (Fig. 2).

IF Part	UNLESS Part	THEN Part
n categorical attributes	n categorical attributes	1 classifying attribute

Fig. 2 The structure of a Classifier

TABLE I
BIRD TRAINING DATA SET

No.	Bird	Wings_broken	Penguin	Dead	Fly
1	Yes	No	No	No	Yes
2	Yes	No	No	No	Yes
3	Yes	No	No	No	Yes
4	Yes	No	No	No	Yes
5	Yes	No	No	No	Yes
6	Yes	No	No	No	Yes
7	Yes	No	No	No	Yes
8	Yes	No	No	No	Yes
9	Yes	No	No	No	Yes
10	Yes	No	No	No	Yes
11	Yes	No	No	No	Yes
12	Yes	No	No	No	Yes
13	Yes	No	No	No	Yes
14	Yes	No	No	No	Yes
15	Yes	No	No	No	Yes
16	Yes	No	No	No	Yes
17	Yes	No	No	No	Yes
18	Yes	No	No	No	Yes
19	Yes	No	No	No	Yes
20	Yes	No	No	No	Yes
21	Yes	No	No	No	Yes
22	Yes	No	No	No	Yes
23	Yes	No	No	No	Yes
24	Yes	No	No	No	Yes
25	Yes	No	No	No	Yes
26	Yes	No	No	No	Yes
27	Yes	No	Yes	No	No
28	Yes	No	No	Yes	No
29	Yes	Yes	No	No	No
30	Yes	No	Yes	Yes	No
31	Yes	Yes	Yes	Yes	No

Each attribute in the rule (IF part or UNLESS part or THEN part) is encoded by single digit (0, 1, 2, 3 ……n where n is the number of values of that attribute). For example, bird attribute can take 3 values as shown below:

Bird	**0**	**Not present**
	1	**Yes**
	2	**No**

Similarly the other attributes are encoded as shown in Fig. 3.

Wings_broken	0	Not present
	1	Yes
	2	No
Penguin	0	Not present
	1	Yes
	2	No
dead	0	Not present
	1	Yes
	2	No
Fly	1	Yes
	2	No

Fig. 3 Individual attribute representation for Bird data set

Consider a randomly generated chromosome (rule set) of size 4 for the bird training data set as shown below in Fig. 4. CPRs can be absurd also as their generation in first iteration is totally random.

Chromosome (Rule Set):

CPR 1: **if** bird = yes **then** fly = no **unless** penguin = no

CPR 2: **if** bird = yes **then** fly = yes **unless** penguin = yes OR dead = yes

CPR 3: **if** bird = yes **then** fly = no **unless** dead = no

CPR 4: **if** bird = yes **then** fly = yes **unless** wings_broken = yes

IF part				**UNLESS** part				**THEN** part
bird	Wings_broken	penguin	dead	bird	Wings_broken	penguin	dead	fly
1	0	0	0	0	0	0	2	2
1	0	0	0	0	1	0	0	1
1	0	0	0	0	0	2	0	2
1	0	0	0	0	0	1	1	1

Fig. 4 Chromosome representation for Bird Dataset

Let us consider the mushroom data set that includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each species is identified as definitely edible, or definitely poisonous.

Number of Instances: 8124

Number of Attributes: 22 (all nominally valued)

Attribute Information: (classes: edible=e, poisonous=p) Representation of the two attributes cap–shape (having four values: fibrous, grooves, scaly, smooth) and scap–surface (having six values: bell, conical, convex, flat, knobbed, sunken) for mushroom data set are shown in Fig. 5. The other details regarding chromosome encoding for the rule set of mushroom data set is similar to that of bird data set.

cap – shape	0	Not present
	1	fibrous
	2	grooves
	3	scaly
	4	smooth

scap-surface	0	Not present
	1	bell
	2	conical
	3	convex
	4	flat
	5	knobbed
	6	sunken

Fig. 5 Individual attributes representation for Mushroom Dataset

The *initial population* of rules within each set of same size is randomly generated with only two restrictions: conditions that can appear in IF part must be true throughout the training data set. Further, the set of attributes in IF part (premise) and the UNLESS part (censor) must be disjoint.

C. Discovery Component

All The *discovery component* basically employs the genetic algorithm for generating CPRs. The heart of most LCS implementations is Genetic Algorithm (GA). Its ability to search efficiently over complex search spaces is known. Most methods of GAs have at least the following elements in common:

- Populations of chromosomes
- Selection according to fitness
- Crossover to produce new offspring
- Random mutation of new offspring

The chromosomes in GAs population typically take the form of bit strings. Each chromosome can be thought of as a point in the search space of candidate solutions. The GA processes populations of chromosomes, successively replacing one such population with another. The GA most often requires a fitness function that assigns a score to each chromosome in the current population. The fitness of a chromosome depends on how well that chromosome solves the problem at hand.

Fitness function:

Once the genetic representation has been defined, the next step is to associate to each solution (chromosome) a value corresponding to the fitness function.

The most difficult and most important concept of evolutionary algorithm is the fitness function. It varies greatly from one type of problem to another. Clearly many criteria are used to quantify the quality or, Fitness, of a rule over the database. Some of these criteria are highly qualitative and in some cases subjective. However in the context of genetic sarch we must formulate a single numerical quantity that encapsulates the desirable features [23]. *Fitness function* is defined as the average of *Accuracy* and *Coverage* as per the details given below.

A CPR of the form **IF** P **THEN** D **UNLESS** C is equivalent to two types of production rules:

Type I. *Rule with censor part present* can be factorized as:

i. $P \Lambda \neg C \rightarrow D$

ii. $P \Lambda C \rightarrow \neg D$

Type II. *Rule without censor*

iii. $P \rightarrow D$

iv. $P \rightarrow \neg D$

Accuracy and Coverage formulae for Rule Type I are:

$A_i = | P \Lambda \neg C \Lambda D | \ / \ | P \Lambda \neg C |$

$C_i = | P \Lambda \neg C \Lambda D | \ / \ | D |$

$A_{ii} = | P \Lambda \ C \Lambda \neg D | \ / \ | P \Lambda \ C |$

$C_{ii} = | P \Lambda \ C \Lambda \neg D | \ / \ | \neg D |$

In the proposed system, the fitness formula for rule type I is:

$Fitness_I = (A_i + C_{i+} A_{ii} + C_{ii}) \ / \ 4$

And for the rules without censor are:

$Fitness_{iii} = (A_{iii} + C_{iii}) / 2$, where

$A_{iii} = | P \Lambda D | \ / | P |$, $C_{iii} = | P \Lambda D | \ / | D |$

and $Fitness_{iv} = (A_{iv} + C_{iv}) / 2$, where

$A_{iv} = | P \Lambda \neg D | / | P |$, $C_{iv} = | P \Lambda \neg D | \ / | \neg D |$

Genetic Operators:

We used conventional genetic operators after appropriate modifications that were necessary for our system requirements to generate CPRs. As we have followed Pittsburgh approach,

crossover and mutation operators are implemented at two levels, i.e. *crossover within a set* (rule level) and *Crossover of two entire sets*; *mutation within a set* (rule level) and *mutation of an entire set*.

Selection is based on the idea that better individuals get higher chance of selection proportional to their fitness.

Crossover is a key operator for natural evolution and is performed at two levels: within a set and between two sets. For *Crossover within a set* of size M, M/2 pairs of rules are randomly chosen one by one using *roulette wheel technique*. In this technique each individual rule is assigned a part of roulette wheel depending on its fitness and wheel is spin for n times to choose n individuals. For the first best pair chosen a random crossover point is generated, and tails are exchanged to create two new offspring. In order to check whether this crossover is valid or not we make the following check:

Offspring produced are legal if the set of attributes present in IF part and the set of attributes present in Unless part are disjoint, IF part is not empty and the offspring produced are distinct from the original rules. In case the offspring are not legal, the crossover process is repeated until we get valid CPRs as offspring.

For one–point *Crossover of two set,* two highly fit sets of size n are chosen randomly following again the roulette wheel technique, then a crossover point, say k, between 1 to n-1 is randomly chosen, and the tails (k^{th} rule to n^{th} rule) of the two sets are swapped.

Mutation is the other way to get new genomes and has been implemented again at two levels *mutation within a set* and *mutation of an entire set***.** For M*utation within a set*, one rule is randomly selected from the set, and one of the 3 parts (IF part, UNLESS part, THEN part) of rule is randomly chosen. Within that chosen part the value of a randomly selected attribute is replaced by one of the values that selected attribute can take.

In *mutation within a set* we mutated 25 % rules within each set and while performing *mutation of entire set* only one rule is randomly chosen and then mutated. It is to be noted that after the mutation if the basic constraints on the CPR are not satisfied then the mutated chromosome is rejected and the mutation process is repeated until a valid mutated CPR is produced.

D. Reinforcement Component

Learning from interaction is a foundational idea underlying nearly all theories of learning and intelligence. Reinforcement learning is defined not by characterizing learning methods, but by characterizing a learning problem. Any method that is well suited to solving that problem, we consider to be a reinforcement learning method. The basic idea is simply to capture the most important aspects of the real problem facing a learning agent interacting with its environment to achieve a goal.

In supervised learning, the true classification of a training case is known to the system, and this information is used by the Reinforcement Component in adjusting the strengths of all classifiers in the system according to the following scheme. First, a Correct Set [C] is created from the classifiers present in a chromosome that are randomly generated by the system, and the remaining classifiers of that chromosome form the set [notC]. This assumes that the decision advocated by the system is correct; if the decision was not correct, then only [notC] is formed. Next, a *tax* is applied to [C], reducing the strength of each classifier in [C] by 10 percent. The purpose of this tax is to inhibit premature convergence and over fitting: the accurate classifiers in [C] at one time step may not be accurate at another. Often, this premature convergence is due to overly general classifiers in the population. The tax helps to "smooth" the asymptotic ascent to an accurate, yet optimally general, population of classifiers. A *reward*, R, is evenly distributed among the classifiers in [C]. R is adjusted so that a higher fraction is apportioned to more general classifiers. The strength of each classifier in [notC] is diminished proportionally by a penalty, typically 50%. The effect of this reward scheme is to exert some degree of selection pressure on the population, such that classifiers are chosen in the discovery component for reproduction based on their strength proportional to that of the other classifiers in the population [13].

After every time step of GA, for all CPRs within each chromosome (ruleset), whose fitness is greater than zero we applied 10% tax by reducing its fitness to avoid over fitting and premature convergence. All correct CPRs with fitness greater than zero are awarded a reward proportional to generality i.e. the more general the rule is, the more reward is awarded to it. The fitness function with reinforcement component becomes:

Fitness = Fitness * TAX + 1 / generality,

where TAX = 0.90 for 10 % tax and generality is count of no. of conditions present in IF part; for rules where censor part is also present generality is proportionally increased depending on number of attributes present in UNLESS part.

III. Experimental Results

Experimental results the bird and mushroom data sets have demonstrated the effectiveness of learning classifier systems approach to generate CPRs and PRs. In each data set the few instances that contained missing values were simply removed. We have assumed that each attribute is categorical, containing discrete data only, in contrast to continuous data such as age, height etc. System was run for 400 generations with 4 chromosomes (rule sets); each rule set of size 40. At the end system maintains set 1 as highly fit set.

Example 1: Let us consider the bird training data set shown in Table I as input to the designed system. Top five rules in the highly fit set i.e. set 1 are summarized below in the Table II.

TABLE II
EXPERIMENTAL RESULT ON BIRD TRAINING SET

SN	Top Five Discovered CPRs and PRs from Highly Fit Set	Accuracy	Coverage	Fitness
1	**if** bird = yes **then** fly = yes	0.838710	1.000000	1.827419
2	**if** bird = yes **then** fly = no	0.161290	1.000000	1.522581
3	**if** bird = yes **then** fly = yes **unless** wings_broken = yes OR penguin = yes OR dead = yes	1.000000	1.000000	1.400000
4	**if** bird = yes **then** fly = yes **unless** penguin = yes OR dead = yes	1.000000	1.000000	1.391667
5	**if** bird = yes **then** fly = yes **unless** wings_broken = yes OR dead = yes	1.000000	1.000000	1.391667

Example 2: After deleting few rows containing missing values, mushroom training data set was given as input to the system and results were observed. Top 3 rules in the highly fit set i.e. set 1 are summarized below in the Table III.

IV. CONCLUSION

In the present work, Learning Classifier Systems approach is proposed for automated discovery of Censored Production Rules (CPRs) that can efficiently handle exceptions and deal with uncertain, incomplete and imprecise knowledge with resource constraints. Systems using Censored Production Rules are free to ignore the exception conditions when time is at premium. The proposed scheme has flexible chromosome representation. Appropriate crossover, mutation and fitness functions have been suggested. The current version of the system handles only categorical attributes and also it cannot cope up with the missing values. But further work towards extending the current system will be to handle continuous valued attributes and missing values is under progress. One of the most important future research directions would be the discovery of Hierarchical Censored Production Rules [21] from large data sets using Learning Classifier Systems Approach.

REFERENCES

[1] Ian H. Witten and Eibe Frank, 'DATA MINING: Practical Machine Learning Tools and Techniques', Second Edition, Morgan Kaufmann Publishers, An imprint of Elsevier.
[2] U.M. Fayyad, G. P. Shapiro and P. Smyth, 'The KDD process for extracting useful knowledge from volumes from data', Communication of ACM, Nov. vol. 39(11), page 27 – 34, 1996.
[3] Pieter Adriaans and Dolf Zantinge, in the book on DATA MINING by Pearson Ed. Publication.
[4] Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press.
[5] Holland, J.H. (1976) Adaptation. In Rosen & Snell (eds) *Progress in Theoretical Biology*, 4.Plenum.
[6] Bernado, E., Llora, X., & Garrell, J. M. (2001). XCS and GALE: a comparative study of two learning classifier systems with six other learning algorithms on classification tasks. In Fourth International Workshop on Learning Classifier Systems - IWLCS-2001 pages 337-341.
[7] Jaume Bacardit and Martin V. Butz, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Passeig Bonanova 8, 08022, Barcelona, jbacardit@salleURL.edu, 'Data Mining in Learning Classifer Systems:Comparing XCS with GAssist'
[8] Holland, J.H., ' Escaping Brittleness: The possibilities of General Purpose Learning Algorithms applied to Parallel rule – based systems', 'Machine Learning: An Artificial Intelligence approach', Vol. II, pages 593 – 623, 1978.
[9] John H. Homes, Pier Luca lanzi, Wolfgang Stolzmann, Stewart W. Wilson, 'Learning Classifier Systems: New Models, Successful Applications.
[10] Kenneth A. De Jong, 'Learning with Genetic Algorithms: An Overview', Machine Learning, 1988.
[11] Michalski, R.S. and Winston, P.H. 1986, 'Variable precision logic', *Artificial Intelligence*, Vol 29, pp 121-146.
[12] Winston, P.H., ' Learning by augmenting rules and accumulating censors', MIT AI Laboratory, AIM – 678, Cambridge, MA, 1982; also in R.S. Michalski, j.G. Carbonell and T. M Mitchell (Eds.), 'Machine learning: An Artificial Intelligence Approach 2 (Morgan Kaufmann, Los Altos, CA, 1986).
[13] Alwyn Barry, John Holmes, and Xavier Llora, 'Data Mining using Learning Classifier Systems' , pages 21 – 23
[14] Pierre Bonelli, Alexandre Parodi, Sandip Sen, and Stewart W. Wilson. 'NEWBOOLE: A Fast GBML System', In *International Conference on Machine Learning*, pages 153–159, San Mateo, California, 1990. Morgan Kaufmann.
[15] John H. Holmes, 'Evolution-Assisted *Discovery of Sentinel Features in Epidemiologic Surveillance'*. PhD thesis, Drexel University, 1996.
[16] John H. Holmes, 'A genetics-based machine learning approach to knowledge discovery in clinical data'. *Journal of the American Medical Informatics Association Supplement*, 1996.
[17] Stewart W. Wilson,' Knowledge Growth in an Artificial Animal'. In Grefenstette [45], pages 16–23. Also appeared in Proceedings of the 4th Yale.
[18] Stewart W. Wilson,' Classifier Systems and the Animat Problem'. *Machine Learning*, 2:199–228, 1987. Also Research Memo RIS-36r, the Rowland Institute for Science, Cambridge, MA, 1986.
[19] John H. Holmes, 'Discovering Risk of Disease with a Learning Classifier System', In Thomas Back, editor, *Proceedings of the 7th International Conference on Genetic Algorithms (ICGA97)*. Morgan Kaufmann, 1997.
[20] Website address of an online data repository; from where Mushroom training dataset and other training data sets are taken: http://www.sgi.com/tech/mlc/db/
[21] K.K. Bharadwaj and N.K. Jain, 'Hierarchical Censored production Rules System', Data and Knowledge Engineering, North Holland, vol. 8, page 19 – 34, 1992.
[22] E. Suzuki, and J.M. Zytkow, 'Unified algorithm for undirected discovery of exception rules', International Journal of Intelligent Systems, vol. 20, page 673 – 691, 2005.
[23] N.J. Radcliffe and P.D. Surry, 'Co – operation through hierarchical competition in genetic data mining', *EPCC – TR94 – 09, 1994.*

TABLE III
EXPERIMENTAL RESULT ON MUSHROOM TRAINING SET

SN	Top Three Discovered CPRs from Highly Fit Set	Accuracy	Coverage	Fitness
1	**if** veil-type = p **then** edible = e **unless** scap-surface = g OR scap-color = p OR odor = p OR gill-attachment = n OR gill-spacing = d OR gill-color = r OR stalk-root = z OR stalk-surface-below-ring = k OR stalk-color-above-ring = y OR veil-color = n OR ring-number = n OR spore-print-color = h OR population = c	0.996982	0.998273	1.388333
2	**if** veil-type = p **then** edible = e **unless** scap-surface = g OR scap-color = p OR odor = p OR gill-attachment = n OR gill-spacing = d OR gill-color = r OR stalk-root = z OR stalk-surface-below-ring = k OR stalk-color-above-ring = y OR veil-color = n OR ring-number = n OR spore-print-color = h	0.996976,	0.998273	1.388094
3	**if** veil-type = p **then** edible = e **unless** scap-surface = g OR odor = p OR gill-attachment = n OR gill-spacing = d OR gill-color = r OR stalk-root = z OR stalk-surface-below-ring = k OR stalk-color-above-ring = y OR ring-number = n OR spore-print-color = h OR population = c	1.000000	1.000000	1.385464