Research Journal of Applied Sciences, Engineering and Technology 4(12): 1701-1707, 2012
ISSN: 2040-7467

Submitted: December 23, 2011 Accepted: January 21, 2012 Published: June 15, 2012

Heuristic Model of Cellular Learning Automata for Fuzzy Rule Extraction

[1]Mohammad Javad Fattahi Hasan Abad and [2]Pouya Derakhshan-Barjoei
[1]Department of Computer Engineering, Ashkezar Branch, Islamic Azad University, Yazd, Iran
[2]Department of Electrical and Computer Engineering, Naein Branch, Islamic Azad University, Isfahan, Iran

Abstract: In this study, a new method has been proposed for rule extraction required for a fuzzy classification system using Cellular Learning Automata Based on Evolutionary Computing (CLA-EC) model. CLA-EC model is an evolutionary algorithm which is a result of the combination of a cellular learning automata with the concepts mentioned in evolutionary computing. It has been shown a higher applicability in optimization fields than CLA and genetic algorithms. Finally, in order to show the applicability of the proposed method, "Iris" standard database for testing purposes has been used.

Key words: Clustering, evolutionary computing, fuzzy rule extraction, learning automata

INTRODUCTION

Fuzzy rule base systems are the same as classical rule base systems that in their "if-then" rules some expressions of fuzzy logic are substituted for some amounts in classical logic. In fact, fuzzy logic in these systems is used as a vehicle to represent the required knowledge for the target problem solving. One of the most important applications of fuzzy sets is their use in fuzzy rule base systems. This feature can consequently result the suitability of these systems for the problems with ambiguity and uncertainty. However, there is a fundamental problem using them and that is the required knowledge or in other words, their required rules for problem solving. Fuzzy input space for rule extraction can be classified in a network form (Jang, 1993; Nozaki *et al*., 1996; Wong and Chen, 2000) or in a dispersed or messy form (Wong and Chen, 1999; Simpson and Fuzzy, 1992). A dynamic classification has been introduced in (Wong and Chen, 2000). In this classification, the input space is firstly classified in a network form, then the center of membership functions are drawn out by means of that classification and using the genetic algorithm, one of the subclasses of the classification made in the previous stage is chosen as the center of membership function and finally fuzzy rule base systems are drawn out. PSO algorithm has been used for fuzzy rule extraction in (Chen, 2006). This mechanism works faster than the genetic algorithm but it has the problem of premature convergence and is sensitive to correct parameters amounts. ANFIS classification system has been introduced in (Jang, 1993). In this system, an input space which contains the membership functions is organized comparatively.

In this study, a new method is introduced for fuzzy rule extraction using CLA-EC model. CLA-EC model is a combination of cellular learning automata model and evolutionary or a heuristic computing. It has been proven that simple CLA-EC model better than CLA or genetic algorithms in optimization problem solving (Masoodi *et al*., 2007; Masoodifar *et al*., 2006).

FORMULATION OF LEARNING AUTOMATA

The aim is to learn to choose the optimal action (i.e., the action with the highest probability of being rewarded) through repeated interaction on the system. If the learning algorithm is chosen properly, then the iterative process of interacting on the environment can be made to result in selection of the optimal action. Learning Automata (Narendra and Thathachar, 1989; Najim and Poznyak, 1994) are adaptive decision-making devices operating on unknown random environments. The Learning Automaton has a finite set of actions and each action has a certain probability (unknown for the automaton) of getting rewarded by the environment of the automaton.

Figure 1 illustrates how a stochastic automaton works in feedback connection with a random environment. Learning Automata can be classified into two main families: fixed structure learning automata and variable structure learning automata (VSLA). In the following, the variable structure learning automata is described. A VSLA is a quintuple $<\alpha, \beta, p, T(\alpha, \beta, p)>$, where α, β, p are an action set with s actions, an environment response set and the probability set p containing s probabilities, each being the probability of performing every action in the current internal automaton state, respectively. The

Corresponding Author: Pouya Derakhshan-Barjoei, Department of Electrical and Computer Engineering, Naein Branch, Islamic Azad University, Isfahan, Iran

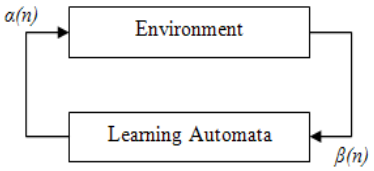

Fig. 1: The interaction between learning automata and environment

Initialize p to [1/s, 1/s,…, 1/s] where s is the number of actions
While not done
Select an action i based on the probability vector p
Evaluate action and return a reinforcement signal β
Update probability vector using learning rule.
End While

Fig. 2: Pseudocode of variable-structure learning automaton

function of T is the reinforcement algorithm, which modifies the action probability vector p with respect to the performed action and received response. Let a VSLA operate in an environment with $\beta = \{0, 1\}$. Let $n \in N$ be the set of nonnegative integers. A general linear schema for updating action probabilities can be represented as follows. Let action *i* be performed at instance n:

$$\begin{aligned} &If \beta(n) = 0, \\ &\quad p_i(n+1) = P_i(n) + a[1 - p_i(n)] \\ &\quad p_j(n+1) = (1-a)\, p_j(n) \qquad \forall_j \ \ j \neq i \end{aligned} \tag{1}$$

$$\begin{aligned} &If \beta(n) = 1, \\ &\quad p_i(n+1) = (1-b)\, p_i(n) \\ &\quad p_i(n+1) = (b/s-1) + (1-b)\, p_i(n) \ \ \forall_j \ \ j \neq i \end{aligned} \tag{2}$$

where *a* and *b* are reward and penalty parameters. When $a = b$, automaton is called L_{RP}. If $b = 0$ and $0<b<<a<1$, the automaton is called L_{RI} and $L_{R\epsilon P}$, respectively. Figure 2 show the working mechanism of learning automata.

Cellular learning automata model: A cellular automaton is an abstract model that consists of large numbers of simple identical components with a local interaction. CA is non-linear dynamical systems which space and time are discrete in. It called cellular, because it is made up cells like points in the lattice or like squares of the checker boards and it is called automata, because it follows a simple rule. One of the models that are used to develop cellular evolutionary algorithm is a Cellular Automaton (CA). The simple components act together to produce complicate patterns of behavior. CA performs complex computation with high degree of efficiency and robustness. It is especially suitable for modeling natural systems that can be described as massive collections of simple object interacting locally with each other. Cellular automaton has not only a simple structure for modeling complex systems, but also it can be implemented easily on SIMD processors. Therefore it has been used in evolutionary computing frequently. Mush literatures are available on cellular automata and its application to evolutionary computing and the interested reader is referred to (Mitchell *et al*., 2000; Wolfram, 1994).

Cellular Learning Automata is a mathematical model for dynamical complex systems that consists of large number of simple components. The simple components, which have learning capabilities, act together to produce complicated behavioral patterns. A CLA is a CA in which learning automaton (multiple learning automaton) is assigned to its every cell. The learning automaton residing in particular cell determines its state (action) on the basis of its action probability vector. Like CA, there is a rule that CLA operate under it. The rule of CLA and the actions selected by neighboring LAs of any particular LA determine the reinforcement signal to the LA residing in that cell. In CLA, the neighboring LAs of any particular LA constitute its local environment, which is nonstationary because it varies as action probability vector of neighboring LAs vary.

The operation of cellular learning automata could be described as follows: At the first step, the internal state of every cell specified. The state of every cell is determined on the basis of action probability vectors of learning automata residing in that cell. The initial value may be chosen on the basis of experience or at random. In the second step, the rule of cellular automata determines the reinforcement signal to each learning automaton residing in that cell. Finally, each learning automaton updates its action probability vector on the basis of supplied reinforcement signal and the chosen action. This process continues until the desired result is obtained (Meybodi *et al*., 2001; Meybodi *et al*., 2003-2004).

Cellular learning automata based on evolutionary computing (CLA-EC): In CLA-EC, similar to other evolutionary algorithms, the parameters of the search space are encoded in the form of genomes. Each genome has two components, model genome and string genome. Model genome is a set of learning automata. The set of actions selected by the set of learning automata determines the second component of the genome called string genome. For each cell, based on a local rule, a reinforcement signal vector is generated and given to the set of learning automata residing in that cell. Each learning automaton based on the received signal update its internal structure according to a learning algorithm. Then, each cell in CLA-EC generates a new string genome and compares its fitness with the fitness of the string genome of the cell. If the fitness of the generated genome is better than the quality of the sting genome of the cell, the generated string genome becomes the string genome of that cell. This process of generating string genome by the

```
Initialize.
While not done do
   For each cell i in CLA do in parallel
      Generate a new string genome
      Evaluate the new string genome
      If f(new string genome)>f(old string genome) then
         Accept the new string genome
      End if
      Select Se cells from neighbors of cell i
      Generate the reinforcement signal vector
      Update LAs of cell i
   End parallel for
End while
```

Fig. 3: Pseudocode of CLA-EC

cells of the CLA-EC is iterated until a termination condition is satisfied. The main issue involved in designing a CLA-EC for a problem is finding a suitable genome representation and fitness function and the parameters of CLA such as the number of cells (population size), topology and the type of the learning automata for each cell.

Evolutionary algorithms as the one described algorithm can be used in any arbitrary finite discrete search space. To simplify the algorithm, we assume that sight search space is a binary finite search space. So the optimization problem can be presented as follows. Assume f: {0,1}m →R be a real function that is to be minimized. In order to use CLA-EC for the optimization function f first a set of learning automata is associated to each cell of CLA-EC. The number of learning automata associated to a cell of CLA-EC is the number bits in the string genome representing points of the search space of f. Each automaton has two actions called action 0 and 1. Then the following steps will be repeated until a termination criterion is met.

- Every automata in a cell i chooses one of its actions using its action probability vector
- Cell i generates a new string genome, new^i, by combining the actions chosen by the learning automata of cell i. The newly generated string genome is obtained by concatenating the actions of the automata (0 or 1) associated to that cell. This section of algorithm is equivalent to learning from previous self-experiences.
- Every cell i computes the fitness value of string genome new^i; if the fitness of this string genome is better than the one in the cell then the new string genome new^i becomes the string genome of that cell. That is:

$$\xi_{t+1}^i = \begin{cases} \xi_t^i & f(\xi_t^i) > f(new_{t+1}^i) \\ new_{t+1}^i & f(\xi_t^i) \le f(new_{t+1}^i) \end{cases} \quad (3)$$

- Se cells of the neighboring cells of the cell i are selected. This Selection is based on the fitness value of the neighboring cells according to truncation strategy. This process is equivalent to mating in the nature. Note that mating in the context of proposed algorithm is not reciprocal, i.e., a cell selects another cell for mating but necessarily vice versa.
- Based on selected neighboring cells a reinforcement vector is generated. This vector becomes the input for the set of learning automata associated to the cell. This section of algorithm is equivalent to learning from experiences of others. Let *Ns(i)* be set selected neighbors of cell i. Define:

$$N_{i,j}(k) = \sum_{l \in N_S(i)} \delta_l(\xi_t^{i,j} = k) \quad (4)$$

where,

$$\delta(\exp) = \begin{cases} 1 & \exp \ is \ true \\ 0 & otherwise \end{cases} \quad (5)$$

$\beta^{i,j}$, the reinforcement signal given to learning automaton j of cell i, is computed as follows:

$$\beta_n^{i,j} = \begin{cases} u(N_{i,j}(1) - N_{i,j}(0)) IF \xi_n^{i,j} = 0 \\ u(N_{i,j}(0) - N_{i,j}(1)) IF \xi_n^{i,j} = 1 \end{cases} \quad (6)$$

where u (.) is a step function. The overall operation of CLA-EC is summarized in the algorithm of Fig. 3. The interested reader is referred to (Rastegar and Meybodi, 2004).

FUZZY CLASSIFICATION STRUCTURE

For this purpose, the structure of the Fuzzy classification system presented in (Wong and Chen, 2000) is used. In this section we will describe it briefly.

Consider an *M*-class classification problem with *n* training patterns $\{X_t, y_t\}$, t = 1, 2, .., n, where $X_t = (x_{t1}, x_{t2}, \ldots, x_{tm})$ is the input vector corresponding to the t-th training pattern and $y_t \in \{1, 2, \ldots, M\}$ represents that the t-th pattern belongs to the y_t-th class. A rule base of a fuzzy system can be expressed as follows:

$$\begin{aligned} &R(j_1, j_2, ..., jm): \\ &IF x_1 is A_{(1, j_1)} and x_2 is A_{(1, j_1)} and ....and \ x_m \\ &is \ A(m, j_m) Then \ X belongs to class \ y(j_1, j_2, ..., j_m) \\ &with \ CF = CF(j_1, j_2, ..., j_m) \end{aligned} \quad (7)$$

$$ji \in \{1, 2, ..., d_i\} \quad , i \in \{1, 2, ..., m\}$$

where X denotes the input vector($x_1, x_2, \ldots, x_m$), d_i denotes the number of fuzzy sets for the input variable x_i, $A_{(i,j)}$ is

the j_i-th fuzzy set of the i-th input variable x_i in the premise part and $y_{(j1,j2,\ldots,jm)} \in \{1, 2, \ldots, M\}$ determines that X belongs to the $y_{(j1, j2, \ldots, jm)}$ -th class and $CF_{(j1, j2 ,\ldots, jm)}$ denotes the grade of certainty of the corresponding fuzzy rule $R_{(j1, j2 ,\ldots, jm)}$. In (Wong and Chen, 2000), the membership function of the fuzzy set $A_{(I, j)}$ is described by:

$$A_{(i,j_i)}(x_i) = \begin{cases} \exp\left(-\left(\dfrac{x_i - c_{(i,j_i)}}{W^l_{(i,ji)}}\right)^2\right) & \text{if } x_i \le c_{(i,j_i)} \\ \exp\left(-\left(\dfrac{x_i - c_{(i,j_i)}}{W^r_{(i,ji)}}\right)^2\right) & \text{if } x_i \le c_{(i,j_i)} \end{cases} \quad (8)$$

where $c_{(1,ji)}$ is the center of the membership function, $W^l_{(i,ji)}$ is the left width value of the membership function and $W^r_{(i,ji)}$ is the right width value of the membership function. Therefore, the shape of the membership function corresponding to the fuzzy set in the premise part is determined by the parameters $\left(c(i,ji), W^l_{(i,ji)}, W^r_{(i,ji)}\right)$. When the t-th input $(x_{t1}, x_{t2}, \ldots, x_{tm})$ is given, the firing strength of the $y_{(j1, j2, \ldots, jm)}$ - th rule is calculated by:

$$\mu^t_{(j1,\ j2,\ldots,\ jm)} = \prod_{i=1}^{m} A_{(i,\ ji)}(x_{ti}) \quad (9)$$

If there are d_i fuzzy sets corresponding to the input variable x_i, the complete fuzzy system will ha $\prod_{i=1}^{m} d_i$ ve rules. On the other hand, the parameters $y_{(j1, j2 ,\ldots, jm)}$ and $CF\, y_{(j1,j2,\ldots,jm)}$ in the consequent part of the corresponding fuzzy rule $R_{(j1, j2, \ldots, jm)}$ are determined in the following procedure.

Procedure 1: Generation of fuzzy rules:

Step 1: Calculate β_{CT} of each class $T \in \{1, 2, \ldots, M\}$ for the fuzzy rule $R_{(j1, j2, \ldots, jm)}$ as follows:

$$\beta_{CT} = \sum_{X_t \in CT} \prod_{i=1}^{m} A_{(i,\ ji)}(x_{ti}) = \sum_{X_t \in CT} \mu^t_{(j1,\ j2,\ldots\ jm)} \quad (10)$$

Step 2: Determine $y_{(j1, j2 ,\ldots, jm)}$ for the fuzzy rule $R_{(j1, j2 ,\ldots, jm)}$ such that

$$\beta_{Cy\,(j1,2\,j,\ldots,jm)} = \max\left\{\beta_{C1}, \beta_{C2}, \ldots, \beta_{CM}\right\} \quad (11)$$

Step 3: Determine the grade of certainty $CF_{(j1, j2 ,\ldots, jm)}$ of the fuzzy rule $R_{(j1, j2 ,\ldots, jm)}$ by

$$CF(j_1, j_2, \ldots j_m) = \frac{(\beta_{cy\,(j_1, j_2, \ldots j_m)} - \beta)}{\sum_{T=1}^{M} \beta_{CT}} \quad (12)$$

where

$$\beta = \sum_{T=1}^{M} \frac{\beta_{CT}}{M-1}$$

In this procedure, the consequent parameter $y_{(j1,j2,\ldots,jm)}$ determined by the largest sum of $\mu^t_{(j1,j2,\ldots,jm)}$ over the M classes.

By the above training procedure, a rule set S is determined as:

$$S = \{ R_{(j1, j2, \ldots, jm)} | j_i = 1, 2, \ldots, d_i, i = 1, 2, \ldots, m\}$$

when a rule set S is given, a new pattern $(x_{t1}, x_{t2}, \ldots, x_{tm})$ is classified by the following procedure based on the fuzzy rules in S.

Procedure 2: Classification of a new pattern:

Step 1: Calculate α_{CT} of each class $T \in \{1, 2, \ldots, M\}$ as follows:

$$\alpha_{CT} = \max\{\mu^t_{(j_1, j_2, \ldots, j_m)}, CF(j_1, j_2, \ldots, j_m) \mid y_{(j_1, j_2, \ldots, j_m)} = TandR_{(j_1, j_2, \ldots, j_m)} \in S\} \quad (13)$$

Step 2: Find Class Y such that

$$\alpha\, CY = \max\{\alpha_{C1}, \alpha_{C2}, \ldots, \alpha_{CM}\} \quad (14)$$

where the value of Y represents the new pattern belongs to the Y-th class.

According to the above description, the parameter set containing of the premise parameters determines a fuzzy classification system. Thus, different premise parameter sets determine different fuzzy classification systems resulting in different performances. In the next section, each individual in CLA-EC is represented as a fuzzy classification system and a method based on CLA-EC is proposed to select an appropriate fuzzy classification system to maximize the number of correctly classified patterns and minimize the number of fuzzy rules.

Using CLA-EC for rule extraction: Suppose our population is a collection of L chromosome, i.e., $P = \{P^1, P^2, \ldots, P^L\}$. When:

$$P^K = \left\{ p^k_{11}, p^k_{12}, \ldots, p^k_{1b_1}, p^k_{21}, p^k_{22}, \ldots, p^k_{2b_2}, \ldots, p^k_{m2}, \ldots, p^k_{mb_m} \right\}$$
$$k \in \{1, 2, \ldots, L\}$$

p^k_{ij} are 0 or 1 and is corresponded by a membership function. Membership function is selected correspondingly with p^k_{ij} ,when a p^k_{ij} is 1, otherwise is not selected. For instance, $p^k_{11}, p^k_{12}, \ldots, p^k_{1b_1}$ indicates corresponding membership functions are characterized by 1st feature. each one of cells in CLA-EC according to Fig. 4 contains a P^k as string genome.

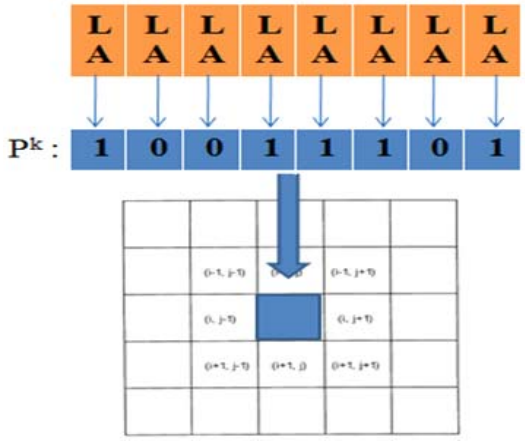

Fig. 4: Each one of cells in CLA-EC contains a P^k as string genome

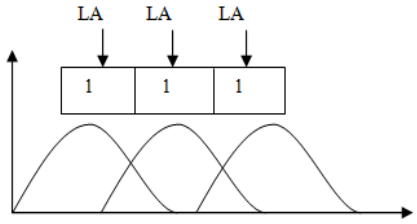

Fig. 5: Allocating learning automata to membership functions and selecting all functions by automata

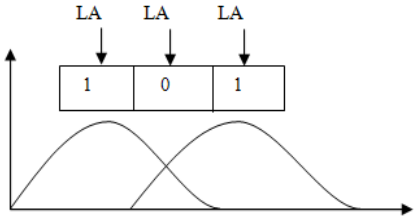

Fig. 6: Allocating learning automata to membership functions and selecting some functions by automata

A learning automata is being responsible for making string genome bits zero or one, or in other words selecting or non-selecting each membership function as shown in Fig. 5 and 6.

If one membership function is not considered, neighboring functions will spread that much to fill the empty space.

Therefore, if automata select 0, the related membership function will not be considered. But the membership function will be considered if automata select 1. Each automaton, considering the operation that it selects and the environment feedback, gradually learns whether its corresponding membership function is useful.

Here, By means of the input vectors $X_t(x_{t1}, x_{t2}, \ldots, x_{tm})$, t = 1, 2, …, n, the range $\left[x_i^{\min}, x_i^{\max}\right]$ of the i-th input variable is determined, where $x_i^{\min} = \min_{t\in\{1,2,\ldots n\}} x_{ti}$ and $x_i^{\max} = \max_{t\in\{1,2,\ldots n\}} x_{ti}$. Therefore, the center position of each membership function with respect to the i-th variable can be determined by (Wong and Chen, 2000):

$$C_{(i,j_i)} = x_i^{\min} + \left(I_{(i,j_i)} - 1\right).\frac{x_i^{\max} - x_i^{\min}}{b_i - 1}, j_i \in \{1, 2, \ldots, d_i\} \quad (15)$$

The notation, $I_{(i,ji)} \in \{1, 2, \ldots, b_i\}$, is defined to represent the index of the substring, $\{p_{i1}^k, p_{i2}^k, \ldots, p_{ib_i}^k\}$ taking a value "1" with respect to the i-th input space. In order to determine the left and right width values of the membership function of the fuzzy set $A_{(i,ji)} i \in \{1, 2, \ldots, m\} j_i \in \{1, 2, \ldots, d_i\}$, we choose a constant $\alpha \in [0,1]$ so that the membership function values of $A_{(i,ji)}(c_{(i,ji-1)})$ and $A_{(i,ji)}(c_{(i,ji+1)})$ are the value of α. That is, from (8):

$$\begin{aligned} A_{(i,j_i)}(c_{(i,j_i)}) &= \exp\left(-\left(\frac{c_{(i,j_i-1)} - c_{(i,j_i)}}{w_{i,j_i}^l}\right)^2\right) = \alpha \\ A_{(i,j_i)}(c_{(i,j_i)}) &= \exp\left(-\left(\frac{c_{(i,j_i-1)} - c_{(i,j_i)}}{w_{i,j_i}^r}\right)^2\right) = \alpha \end{aligned} \quad (16)$$

Therefore, the left and right width values of each membership function with respect to the i-th input variable can be, respectively calculated by:

$$\begin{aligned} w_{(i,j_i)}^l &= \sqrt{\frac{(c_{(i,j_i-1)} - c_{(i,j_i)})2}{-1n(\alpha)}} \\ w_{(i,j_i)}^r &= \sqrt{\frac{(c_{(i,j_i-1)} - c_{(i,j_i)})2}{-1n(\alpha)}} \end{aligned} \quad (17)$$

where $c_{(i,0)} = x_i^{\min}, c_{(i,d_i+1)} = x_i^{\max}$ and $\alpha \in [0, 1]$ is a constant to control the overlapping of two adjacent membership functions. By means of the preceding process, the number and shapes of membership functions in the premise part are constructed to partition the input space and the number of fuzzy rules is determined.

An individual determines a fuzzy classification system. In order to construct an appropriate fuzzy classification system to maximize the number of the correctly classified patterns and minimize the number of fuzzy rules, the fitness function is defined as follows:

$$F(p^k) = W_{NCP}.NCP(S(p^k)) - W_s.\left|S(p^k)\right| \quad (18)$$

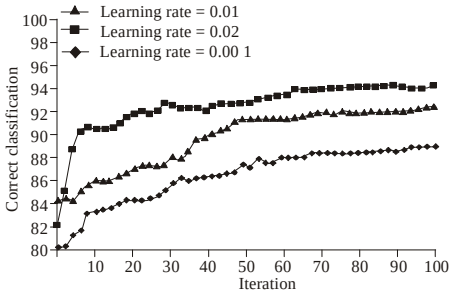

Fig. 7: Correct classification percent

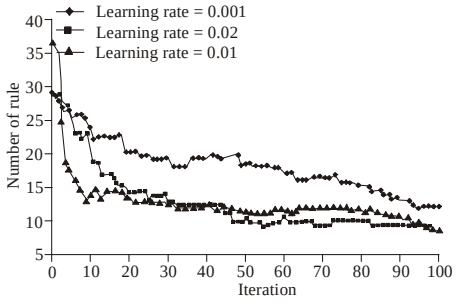

Fig. 8: Members of rule extraction graph

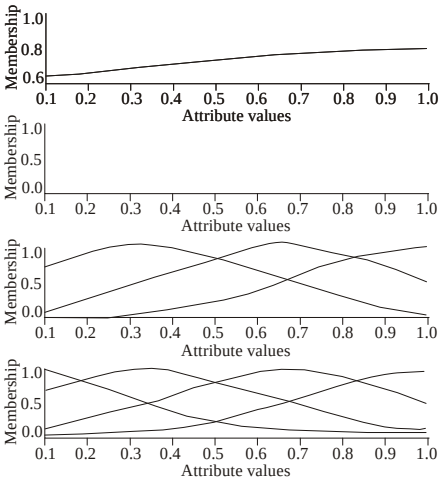

Fig. 9: Graph of created membership function with learning rate of 0.001

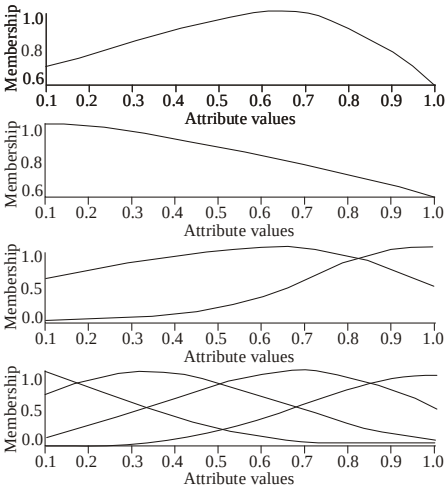

Fig. 10: Graph of created membership function with learning rate of 0.01

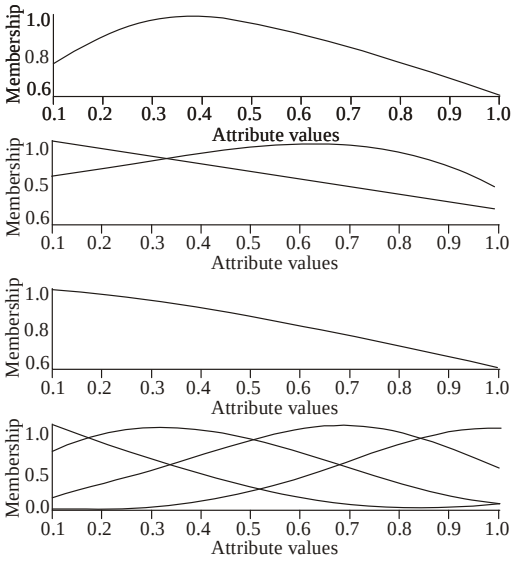

Fig. 11: Graph of created membership function with learning rate of 0.02

where $S(p^k)$ is a rule set corresponding to the individual p^k, $NCP(S(p^k))$ denotes the number of correctly classified patterns by $S(p^k)$, $|S(p^k)|$ denotes the number of fuzzy rules by $S(p^k)$, W_{NCP} is the weight of $NCP(S(p^k))$ and W_S is the weight of $|S(p^k)|$. In general, the classification power of a classification system is more important than its compactness. Therefore, the weights should be specified as $0<W_S<<W_{NCP}$. In this way, the fitness function will guide the individual to find an appropriate fuzzy classification system to satisfy the desired objective. That is, the final selected fuzzy classification system using CLA-EC can simultaneously achieve the combinatorial objectives for maximizing the number of correctly classified patterns and minimize the number of fuzzy rules.

Experimental experience: A linear CLA-EC with neighboring radius of 2 and variable learning rate has been used in experiments and in each cell learning automata is L_{RI} type. We have used Iris data base for doing experiments. This data base includes 150 data which each data owns 4 features and belongs to one of these 3 classes: Iris Setosa, Iris Versicolour and Iris Virginica. There are 50 data in each class. One of these

Table 1: Values of parameters used in experiments

Values	Parameters
Number of cells	20
The length of bits for each input variable (b_i)	4
Neighboring radius	2
Number of selected cells	2
Overlapping control constant (α)	0.5
W_{ncp} , W_s	0.3,0.7
Learning rate	0.001,0.01,0.02
Number of generations (N)	100
Number of repeat	20

Table 2: Comparing CLA-EC efficiency with other algorithms used in classification

Algorithms	Number of rules	Number of correctly classifiend patterns (%)
ANFIS	81	99.5
Pruning	28	93.3
Multi-rule-table	597	94.3
PSO-based	5	96.8
GA-based	10	90.67
LA-based	11	93.63
CLA-EC, LR = 0.001	12	89.75
CLA-EC, LR = 0.01	8	92.31
CLA-EC, LR = 0.02	8	94.48

classes is clearly separated from others linearly and the other 2 classes can be distinguished from each other non-linearly. The rest of parameters are shown in Table 1.

Results of experiments using CLA-EC model with learning rate of 0.001, 0.01 and 0.02, respectively are shown in Fig. 7 to 11. As it is observed, increasing learning rate in the model improves algorithm efficiency and increases its speed. Appointing learning rate parameter value in the model is based on the problems dealing with classification we can assume higher value for the parameter to speed up the algorithm. If data's structures and dimensions that are to classify are not very complex. But if their structures and dimensions are complex, we must decrease learning rate, other algorithm get closer answer more carefully.

Table 2 compares CLA-EC model efficiency and other algorithms used in Iris data base classification. And results indicate the model efficiency over other similar models and methods.

CONCLUSION

In this study, a new method has been proposed for rule extraction using Cellular Learning Automata Based on Evolutionary Computing (CLA-EC) model. The CLA-EC has a number of properties that make it superior over other evolutionary models. A highly degree of diversity is apparent in the early generations created by having the probabilities initially random and only slightly biased in the early iteration. In other hand with respect to the fact that interactions between cells (genomes) are local the probability of stuck in local optima can be decreased. The empirical results have shown that the proposed algorithm is an available and effective approach for rule extraction problem.

REFERENCES

Chen, C.C., 2006. Design of PSO-based fuzzy classification systems. Tamkang J. Sci. Eng., 9(1): 63-70.

Jang, J.S., 1993. ANFIS: Adaptive-network-based fuzzy inference systems. IEEE T. Syst. Man Cyb., 23: 665-685.

Masoodi, B., M.R. Meybodi and M. Hashemi, 2007. Cooperative CLA-EC. Proceedings of 12th Annual CSI Computer Conference of Iran, Shahid Beheshti University, Tehran, Iran, pp: 558-559, Feb. 20-22.

Masoodifar, B., M.R. Meybodi and R. Rastegar, 2006. Asynchronous CLA-EC, Proceedings of 11th Annual CSI Computer Conference of Iran, Fundamental Science Research Center (IPM), Computer Science Research Lab., Tehran, Iran, pp: 447-458, Jan. 24-26.

Meybodi, M.R., H. Beigy and M. Taherkhani, 2001. Cellular Learning Automata, Proceedings of 6th Annual International Computer Society of Iran Computer Conference CSICC2001, Isfahan, Iran, pp: 153-163.

Meybodi, M.R., H. Beigy and M. Taherkhani, 2003-2004. Cellular Learning Automata and Its Applications, Journal of Science and Technology, University of Sharif, No. 25, pp: 54-77, Autumn/Winter.

Mitchell, M., P. James, E. Crutch and D. Rajarshi, 2000. Evolving Cellular Automata with Genetic Algorithms. Proceedings of the First International Conference on Evolutionary Computation and Its Applications (EvCA'96) Moscow, Russian Academy of Sciences, Russia.

Najim, K. and A.S. Poznyak, 1994. Learning Automata: Theory and Application, Tarrytown. Elsevier Science Publishing Ltd., New York.

Narendra, K.S. and M.A.L. Thathachar, 1989. Learning Automata: An Introduction. Prentice Hall, Englewood Cliffs, N.J.

Nozaki, K., H. Ishibuchi and H. Tanaka, 1996. Adaptive fuzzy rule-based classification systems. IEEE T. Fuzzy Syst., 4(3): 238-250.

Rastegar, R. and M.R. Meybodi, 2004. A New Evolutionary Computing Model based on Cellular Learning Automata, Proceedings of 2004 IEEE Conference on Cybernetics and Intelligent Systems (CIS), pp: 433-438, Singapore 1-3 Dec.

Simpson, P. and K. Fuzzy, 1992. Min-max neural networks-part 1: Classification. IEEE T. Neural Networks, 3: 7760-786.

Wolfram, S., 1994. Cellular Automata and Complexity. Perseus Books Group, United States.

Wong, C.C. and C.C. Chen, 1999. A hybrid clustering and gradient descent approach for fuzzy modeling. IEEE T. Syst. Man Cy. B., 29: 686-693.

Wong, C.C. and C.C. Chen, 2000. A GA-Based method for constructing fuzzy systems directly from numerical data. IEEE T. Syst. Man Cy. B., 30: 904-911.