Available online at www.sciencedirect.com

ScienceDirect

Pattern Recognition Letters 29 (2008) 40–48

Pattern Recognition Letters

www.elsevier.com/locate/patrec

Learning automata based classifier

Seyed-Hamid Zahiri [*]

Department of Electrical Engineering, Faculty of Engineering, Birjand University, P.O. Box 97175-376, Birjand, Iran

Received 22 April 2006; received in revised form 6 June 2007
Available online 24 August 2007

Communicated by M. Kamel

Abstract

In this paper a new classifier has been designed based on the learning automata. This classifier can efficiently approximate the decision hyperplanes in the feature space without need to know the class distributions and the a priori probabilities. The performance of the proposed classifier has been tested on different kinds of benchmarks with nonlinear, overlapping class boundaries and different feature space dimensions. Extensive experimental results on these data sets are provided to show that the performance of the proposed classifier is comparable to, sometimes better than multi-layer perceptron, *k*-nearest neighbor classifier, genetic classifier, and particle swarm classifier. Also the comparative results are provided to show the effectiveness of the proposed method in comparison to similar researches. Furthermore the effect of the number of training points on the performance of the designed classifier is investigated. It is found that as the number of training data increases, the performance of the classifier tends to the performance of Bayes classifier which is an optimal one.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Learning automata; Decision hyperplanes; Data classification; Pattern recognition

1. Introduction

Learning automata (LA) is referred to an automaton acting in an unknown random environment which improves its performance to obtain an optimal action. An action is applied to a random environment and the random environment evaluates the applied action and gives fitness to the selected action of automata. The response of the environment is used by automata to select its next action. This procedure is continued to reach the optimal action.

LA has been used in several tasks of engineering problems (for example graph partitioning (Oommen and de St.Criox, 1996) adaptive control (Zeng et al., 2000), signal processing (Tang and Mars, 1993), and power systems (Wu, 1995)).

Also there are researches which explain some applications of LA in pattern recognition tasks. For example Sastry and Thathachar proposed algorithms based on teams of learning automata for pattern classification (Sastry and Thathachar, 1999). In their algorithms, the users should know the optimal parameter representation for the discriminant functions to reach a good recognition score. They also used a three-layer network consisting of teams of automata for pattern classification (Sastry and Thathachar, 1999; Thathachar and Sastry, 2002). They tested a network consisted of nine first layer and three second-layer units on the two-class version of the Iris data set (Sastry and Thathachar, 1999; Thathachar and Sastry, 2002).

Recently some effective algorithms have been proposed for multimodal complex function optimization based on the LA (e.g. Zeng and Liu, 2005; Beygi and Meybodi, 2006). It was shown experimentally that the performance of these optimization algorithms is comparable to or better than the genetic algorithm (GA) (Zeng and Liu, 2005). On the other hand, the powerfulness of GA and particle swarm optimization (PSO) algorithm in the search space (here, feature space) motivated the authors to present novel and effective classifiers which need no important prior knowledge. For example, Bandyopadhyay et al. (1999) proposed

[*] Tel.: +98 561 2227044; fax: +98 561 2227795.
E-mail address: hzahiri@birjand.ac.ir

0167-8655/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2007.08.011

a genetic algorithm based classifier (called GA-classifier). Also Zahiri and Seyedin (2007) introduced another classifier based on the particle swarm optimization algorithm (called PS-classifier). Now, it is possible to propose another effective classifier, using function optimization algorithms based on the LA. This is named LA-classifier.

In this article a LA-classifier is described which utilizes LA for function optimization to find the decision hyperplanes between the different classes without any important prior knowledge. In the proposed algorithm the feature space is separated into individual regions with predefined number of hyperplanes and it is not necessary to know the optimal parametric representation of discriminant functions. Note that a characteristic feature of the LA-classifier is that it utilizes the decision boundary for performing classification. This is in contrary to the conventional pattern-classification techniques where the decision boundaries are obtained as a consequence of the decision-making process.

Moreover, in this paper, the effect of the number of training points on the performance of the LA-classifier has been considered by comparison of its performance with Baysian classifier, which is an optimal classifier.

Five common benchmarks have been considered for comparative experimental results to investigate the effectiveness of the proposed classifier. Those are Iris, Wine, Glass, Appendicitis, and Cancer data classification with different feature space dimensions (4–13), nonlinear and overlapping class boundaries.

Performance evaluation of the designed classifier on aforesaid data sets and its comparison with different kinds of conventional and novel classifiers (*k*-nearest neighbor, multi-layer perceptron, GA-classifier, and PS-classifier) show that the averages of recognition scores of the designed LA-classifier are better than or comparable to other aforementioned classifiers. Also the comparison between the LA network classifier which was proposed by Sastry and Thathachar (1999) and Thathachar and Sastry (2002), and our algorithm shows a considerable improvement for LA-classifier.

Furthermore, it is found that as the number of training points increases, the performance of the new classifier tends to the Bayes classifier which is an optimal classifier when the class distribution and the prior probabilities are known.

In this paper, Section 2 describes the basic principles of learning automata. Learning automata based classifier is introduced in Section 3. Section 4 considers implementation of the classifier and experimental results on the five aforesaid pattern recognition problems and comparison with existing methods. Finally, conclusion and discussion is presented in Section 5.

2. Principles of learning automata

Learning automata (LA) are adaptive decision-making units that can learn to choose the optimal function from a set of actions by interaction with an environment (search space). In the other words a learning automaton is a stochastic automaton in feedback connection with a random environment. Each learning automaton is characterized by a set of internal states, input actions, state probability distributions, a reinforcement scheme, and is connected in feedback loop to the environment as shown in Fig. 1.

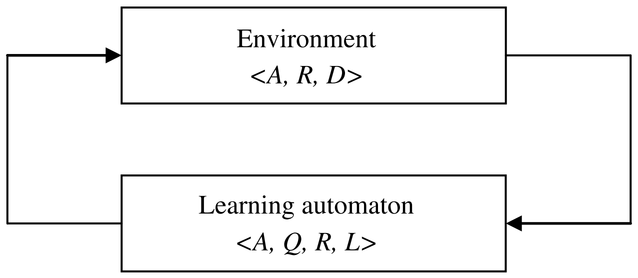


Fig. 1. Automaton operating in the environment.

One main advantage of learning automaton is that it needs no important knowledge of the environment in which it operates or any analytical knowledge of the function to be optimized.

A finite learning automata is generally defined by $\langle A,Q,R,L\rangle$ and the environment by $\langle A,R,D\rangle$, where $A = \{\alpha_1,\alpha_2,\ldots,\alpha_r\}$ is defined as all actions of the automaton where $\alpha(k)$ is the automaton at the instant k and $\alpha(k) \in A$ for $k = 0,1,2\ldots$ and r is the total number of actions. In fact A is the set of outputs of the automaton and it is also the set of inputs to the environment. R is the domain of responses from the environment. Let $\beta(k)$ denote the response received by the automaton at instant k where $\beta(k) \in R\ \forall k$. $\beta(k)$ is the output of the environment and it is also the input to the automaton. $D = \{d_1,d_2,\ldots,d_r\}$ is the set of reward probabilities, where $d_i(k) = E[\beta(k)|\alpha(k) = \alpha_i]$. The reward probabilities are unknown for the automaton. Q is the state of the automaton defined by $Q(k) = [P(k),\widehat{D}(k)]$ where $P(k) = [p_1(k),p_2(k),\ldots,p_r(k)]$ is the action probability vector $(0 \leqslant p_i(k) \leqslant 1$ and $\sum_{i=1}^{r} p_i(k) = 1,\ \forall k)$ and $\widehat{D}(k) = [\widehat{d}_1(k),\widehat{d}_2(k),\ldots,\widehat{d}_r(k)]$ is the vector of estimates of the reward probabilities at instant k. L is the learning algorithm or the reinforcement scheme which is used by the automaton in order to update its state. In fact $Q(k+1) = L(Q(k),\alpha(k),\beta(k))$.

At each instant k, the automaton selects an action $\alpha(k)$ from the set of all actions A. This selection depends on the current action probability vector $P(k)$. The selected action $\alpha(k)$ becomes input to the environment and the environment gives the input of the automaton a random response $\beta(k)$ whose expected value is $d_i(k)$ if $\alpha(k) = \alpha_i$. Next, the automaton calculates $Q(k+1)$ using the reinforcement scheme L.

This procedure is continued until the optimal action to the environment is found. We denote the optimal action as α_m with expected value of $d_m = \mathrm{Max}\{d_i\}$ for all $i = 1,2,\ldots,r$. It is desired that the action probability corresponding to α_m tends to unity as the time k goes to infinity.

Two main groups of LA are finite action set LA (FALA) and continuous action set LA (CALA). If $r \to \infty$ the afore described finite action set LA is changed to CALA. Several kinds of FALA and CALA have been reported in the literature (e.g. Thathachar and Sastry, 2002). But they have common principles as explained above.

3. Learning automata based classifier

Learning automata based classifier (LA-classifier) is established on two major parts including decision hyperplanes and function optimization using LA which are appeared in details below.

3.1. Decision hyperplanes

A general hyperplane is in the form:

$$d(X) = w_1x_1 + w_2x_2 + \cdots + w_nx_n + w_{n+1} \qquad (1)$$

where $X = (x_1, x_2, \ldots, x_n, 1)$ and $W = (w_1, w_2, \ldots, w_n, w_{n+1})'$ are called the augmented feature and weight vector respectively. n is the feature space dimension.

In a general case, there are a number of hyperplanes ($\log_2^M$, is the minimum value, where M is the number of classes) that separate the feature space to different regions, where each region distinguishes an individual class. Fig. 2 shows a simple example containing six classes encoded by three decision lines. In this figure, IR denotes the indeterminate region.

Some special cases are shown in Fig. 3 (Tou and Gonzalez, 1992).

In Fig. 3a the decision of classifier depends on the sign of each decision line. It means that:

$$X \in C_i \quad \text{if } d_i(X) = W_i'X > 0, \quad i = 1, 2, \ldots, M, \qquad (2)$$

where $d_i(X)$ can be the ith hyperplane, C_i is the ith class and W_i is the weight vector for the ith hyperplane.

Fig. 3b shows the case that each pattern class is separable from other classes by a distinct decision surface, that is, the classes are pairwise separable, $M*(M-1)/2$ number of hyperplanes are needed and no indeterminate regions

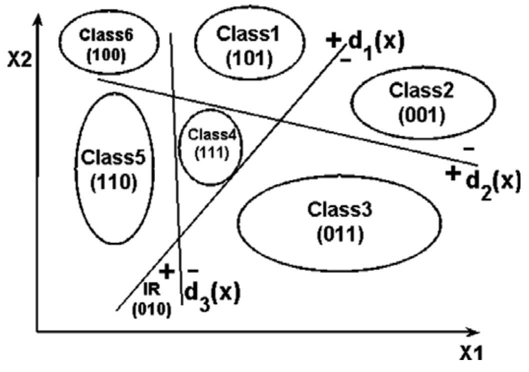


Fig. 2. Each region can identify an individual class by its code, which obtained from the sign of linear decision functions.

(IR) exist. In this case the decision functions are of the form:

$$\begin{aligned} d_{ij}(X) &= d_i(X) - d_j(X) = W_i'X - W_j'X \\ &= (W_i - W_j)'X = W_{ij}'X. \end{aligned} \qquad (3)$$

So the classifier rule will be:

$$X \in C_i \quad \text{if } d_i(X) - d_j(X) > 0 \quad \text{for all } j \neq i$$

or

$$X \in C_i \quad \text{if } d_i(X) > d_j(X) \quad \text{for all } j \neq i. \qquad (4)$$

The LA-classifier must find W_i $(i = 1, 2, \ldots, H)$ in solution space, where H is the number of decision hyperplanes and should be predefined by user.

3.2. Function optimization algorithms based on the learning automata

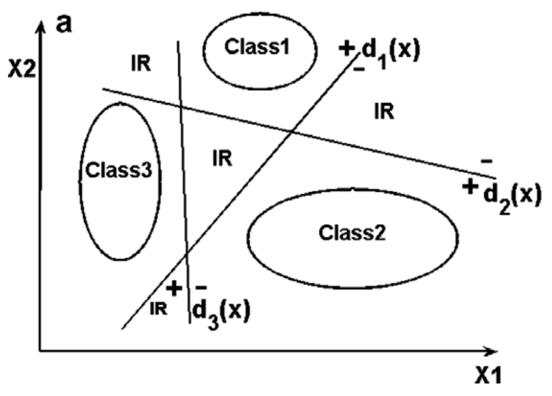
As mentioned in Section 1 the proposed classifier has been established on the basis of function optimization using LA. We define a function on the weight vectors variable of $\underline{W}$ as below:

$$f(\underline{W}) = \text{Miss}. \qquad (5)$$

In Eq. (5) $\underline{W} = \{W_1, W_2, \ldots, W_H\}$ is the set of all H hyperplanes weigh vectors and Miss is the number of misclassified data points by $\underline{W}$. Thus $f(\underline{W})$ returns the total number of misclassified training points by a set of weight vectors of $\underline{W}$. Obviously by minimizing $f(\underline{W})$ it is possible to calculate the H hyperplanes. Utilizing a powerful function optimization algorithm based on the learning automata leads us to design LA-classifier.

Many kinds of gradient, heuristic, evolutionary and swarm intelligence based algorithms have been reported in the literature. Among them genetic algorithm (GA) and particle swarm optimization (PSO) are two powerful and famous optimization algorithms such that two novel evolutionary and swarm intelligence based classifiers (i.e. GA-classifier and PS-classifier) were designed based on them (Bandyopadhyay et al., 1999; Zahiri and Seyedin, 2007). Both GA-classifier and PS-classifier search the feature space to obtain a set of hyperplanes minimizing the fitness function defined by Eq. (5).

But related to LA area of researches, over the years, many studies on the LA based function optimization techniques were reported. A stochastic automata model was adopted by Shapiro and Narendra (1969) to find an optimal solution for multimodal performance criteria. Thathachar and Sastry (1985) proposed Pursuit algorithm to improve the convergence rate of LA based function optimization. Also an automata model was proposed by Oommen and Lanctôt (1990), using discretized Pursuit algorithm. Obaidat et al. (2003) developed an algorithm for fast convergence of learning automata. Beygi and Meybodi (2006) proposed a continuous action-set automaton for function optimization. They studied the convergence properties of



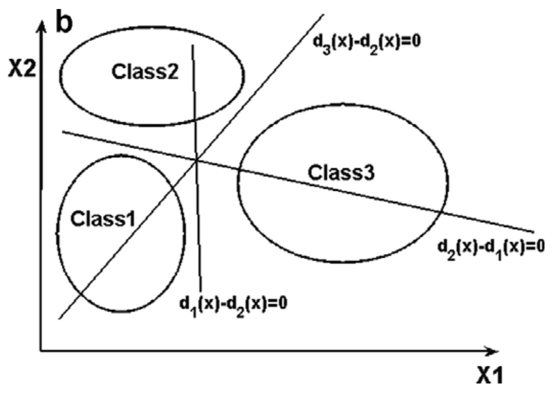


Fig. 3. (a) Each class is separable from other classes and (b) the classes are pairwise separable.

their algorithm theoretically and experimentally. Also Zeng and Liu (2005) presented a method for optimizing continuous functions using LA. Their method enhances local search in interesting regions or intervals and reduces the whole searching space by removing useless regions.

We used the optimization algorithm introduced by Zeng and Liu (2005) for designing the proposed classifier because they showed experimentally that their proposed LA based optimization algorithm works even better than genetic algorithm.

In this algorithm at first the solution space is uniformly divided into r hypercubes each corresponding to one action of the learning automaton. Then using continuous Pursuit algorithm the action probabilities and estimates of reward probabilities are updated at each period by calculating the function value of a randomly selected sample corresponding to the current action. If the estimate of a reward probability is smaller than a predefined threshold, the corresponding hypercube is then evaluated according to the samples whose function values have been calculated. If both the mean value and the variance of these function values are small enough, this hypercube is considered as stable and useless. Then, this hypercube is removed and the optimization continues with the remaining $r-1$ hypercubes. Otherwise, this hypercube is considered as unstable and the rising and falling (pinks and valleys) of the function are estimated in this hypercube from the samples inside it. Next, this hypercube is divided into a number of sub-hypercubes each only containing ascending or descending samples and the original hypercube is replaced by the best rewarded sub-hypercube. The other sub-hypercubes are considered as useless and then removed. In this way, the number of actions is unchanged. This procedure is repeated until a predefined precision condition is satisfied. Then, original hypercubes are either removed or converge to several values in which are included a quasi-global optimum, i.e. a solution whose function value is rather close to that of the global optimum. Like other stochastic optimization algorithms, this method aims at finding a compromise between exploration and exploitation, i.e. converging to the nearest local optima and exploring the function behavior in order to discover global optimal region.

3.3. The structure of LA-classifier

According to above descriptions the structure of LA-classifier based on minimizing $f(\underline{W})$ is as follows:

Step 1: Initialization of internal parameters

Internal parameters are:
r: Number of hypercubes which divide the feature space
δ: Threshold of action probabilities
s: Normalized factor of convergence
ε: Error band

Step 2: Divide the feature space into r hypercubes

Each hypercube is supposed as an action denoted by α_i, $i \in \{1, 2, \ldots, r\}$.
Corresponding to each action there are some parameters introduced as below:
$\eta_i(n)$: Total reward obtained by the action α_i until nth sampling instant.
$z_i(n)$: Number of times the action α_i is chosen until nth sampling instant.

$$\xi_i(n) = \frac{\eta_i(n)}{z_i(n)},$$

$$\xi_m(n) = \mathrm{Max}_i\{\xi_i(n)\},$$

$$\xi_l(n) = \mathrm{Min}_i\{\xi_i(n)\}.$$

$p(n)$: Action probability distribution of α_i at nth sampling instant with initial value of $\frac{1}{r}$.

Step 3: Search loop

Repeat

– Pick up an action $\alpha(n) = \alpha_i(n)$ according to $p(n)$.
– Randomly select a set of decision weight vector of H hyperplanes in the form of $\underline{W} = \{W_1, W_2, \ldots, W_H\}$ from the hypercube corresponding to α_i.
– Calculate $f(\underline{W})$.
– Update $\xi_i(n)$ as follows:
  If $\alpha(n) = \alpha_i$, Then

$$\eta_i(n+1) = \eta_i(n) + \frac{T - f(\underline{W})}{T},$$

where T is the total number of training data permitting to normalize the estimates of reward probabilities to the interval $[0,1]$

$$z_i(n+1) = z_i(n) + 1,$$
$$\xi_i(n+1) = \frac{\eta_i(n+1)}{z_i(n+1)}.$$

For all $j \neq i$

$$\eta_j(n+1) = \eta_j(n),$$
$$z_j(n+1) = z_j(n),$$
$$\xi_j(n+1) = \xi_j(n).$$

– Update $p(n)$ as follows:

$$p(n+1) = (1 - s * \xi_m(n)) * p(n) + s * \xi_m(n) * e_m,$$

;(e_m is a r-dimensional vector with mth component unity and all others zero)

– If $p_l(n) = \mathrm{Min}_i\{p_i(n)\} < \delta$, Then
  Go to the next step,
– Else

$n = n + 1$,

End Repeat;

At this step, the action probability distribution $p(n)$ and estimates of reward probabilities $\xi_i(n)$ are updated using continuous pursuit algorithm. Evidently, the values of $\xi_i(n)$ vary proportionally with the current value of $f(\underline{W})$ and the values of $p(n)$ vary with the best reward probability estimate $\xi_m(n)$. If $\xi_m(n)$ is strongly rewarded, its values are close to 1 and then the updated distribution of action probabilities enhances the selection of the corresponding action α_m for the following instants. If the estimates of all reward probabilities are strongly penalized, their values are close to 0 and the distribution of action probabilities maintains almost unchanged for following instants.

Step 4: Enhancing the search process in *l*th hypercube

At this step the search process is enhanced in the l hypercube by dividing it into r sub-hypercubes corresponding to r new actions αl_i's $(i = 1, \ldots, r)$:

– Recursively calculate the corresponding $pl_i(n_1)$'s and reward probabilities $\xi l_i(n_1)$ using the equations at Step 2 and Step 3 until the following condition holds:

$$pl_q(n_1) = \min\{pl_i(n_1)\} < \delta_1, \quad 0 < \delta < \delta_1 < 1.$$

– Calculate the average reward Md and the variance Var:

$$M_\xi = \frac{\sum_{k=1}^{r} \xi l_k(n_1)}{r},$$
$$\mathrm{Var} = \max_i\{|\xi l_i(n_1) - M_\xi|\}.$$

– If Var $< \varepsilon$ and $\xi l_m(n_1) < \xi_m(n)$, Then
  Remove the lth interval, (it is considered as stable and useless for further search).
  Subdivide a remaining hypercube into two sub-hypercubes each corresponding to a new action (the total number of actions is not changed).
  Update $p(n)$,
  Go to Step 3;
– Else
  Go to Step 5;

Step 5:

– Calculate $\gamma_i = \xi l_{i+1}(n_1) - \xi l_i(n_1), \quad i = 1, \ldots, r-1$ (lth hypercube is considered as unstable).
– Estimate from γ_i's the number of pinks and valleys of $f(\underline{W})$ in the lth hypercube, denoted by k.
– Redevide the lth hypercube into k sub-hypercubes each corresponding to one pink or valley.
– Generate k new actions αlm_i, $1 \leqslant i \leqslant k$, and repeat Step 5 for calculating the action probabilities and reward probabilities (let αlm be the action corresponding to the biggest value of the reward probabilities).
– Replace the lth hypercube.
– If the hypervolume of this hypercube is smaller than εthen go to Step 6, Else go to Step 3.

Step 6:

– Adding the midpoint of the lth hypercube to a list and remove this hypercube.
– If the number of remaining hypercubes is zero, then select the global optimum from the list and stop;
– Else, subdivide a remaining hypercube into two sub-hypercubes each corresponding to a new action and go to Step 3.

Briefly, by executing these steps, some of the hypercubes with better action and reward probabilities are selected and then the search process is enhanced in them. Each of them is divided into r sub-hypercubes corresponding to r new actions and the search loop is continued to reach the value of $\underline{W}$ (weight vectors of the decision hyperplanes) which minimizes $f(\underline{W})$. In general the search loop continues for a predefined number of iterations.

4. Implementation and results

Extensive empirical results are provided in this section for demonstrating the effectiveness of the designed LA-classifier. Five pattern recognition benchmarks with different augmented feature vectors dimensions (5–14), nonlinear, overlapping class boundaries, and different number of reference classes (2–6) were used to evaluate the performance of the LA-classifier. Also the performance of the proposed classifier is compared to several kinds of conventional, evolutionary and optimal classifiers. k-nearest neighbor (k-NN), multi-layer perceptron (MLP) are considered as two conventional and well-known classifiers.

GA-classifier and PS-classifier are two novel classifiers designed based on the evolutionary and swarm intelligence algorithms respectively (Bandyopadhyay et al., 1999; Zahiri and Seyedin, 2007). Also three-layer network consisting of teams of automata (named AN-classifier) as proposed by Sastry and Thathachar (1999) and Thathachar and Sastry (2002) has been considered for comparative results. Finally some experimental results are provided to show the effect of the number of training points on the performance of the proposed classifier.

This section is divided into three parts. The description of the data sets is given in the first part. The comparison results with selected conventional and novel classifiers are presented in the second part. Finally, the third part shows the effect of the number of training points on the performance of the LA-classifier by comparing its performance with that of the Bayes classifier which is an optimal classifier when the class distribution and the prior probabilities are known.

4.1. Data sets

Iris data:[1] The Iris data contains 50 measurements of four features from of each three species Iris setosa, Iris versicolor, and Iris virginica. Features are sepal length, sepal width, petal length and petal width.

Wine data:[1] The Wine data contains the chemical analysis of wines grown in the same region in Italy but derived from different cultivars. The 13 continuous attributes are available for classification. The number of classes is three and the number of instances in each class is 59, 71 and 48 respectively.

Glass data:[1] The Glass data consists of 214 samples with nine continuous attributes from six classes.

*Appendicitis data:*This data set consists of 106 samples with seven attributes from two classes.

Cancer data: This breast cancer database, obtained from the University of Wisconsin Hospital, Madison, has 683 breast mass samples belonging to two classes *Benign* and *Malignant,* in a nine-dimensional feature space.[2]

4.2. Comparison with existing conventional and novel classifiers

The performance of the proposed LA-classifier is compared to the performance of k-nearest neighbor (k-NN), multi-layer perceptron (MLP), genetic algorithm based classifier (GA-classifier), particle swarm classifier (PS-classifier), and network consisting of teams of automata (AN-classifier). MLP and k-NN are two well-known conventional classifiers. GA-classifier and PS-classifier are two new classifiers constructed based on the GA and PSO respectively. AN-classifier is a method for data classification based on the LA proposed in (Sastry and Thathachar, 1999; Thathachar and Sastry, 2002).

To construct the LA-classifier the search space is separated to individual regions in each problem using the relation $r = z^q$, where r is the number of individual hypercubes, q is the feature space dimension, and z is the number of divided hypercubes for each variable (we take $z = 3$ for each problem). The maximum number of iterations is set to 5000.

k-NN classifier is executed taking k equal to $\sqrt{T}$, where T is the number of training samples (it is known that as the number of training patterns T goes to infinity if the value of k and k/T can be made to approach infinity and 0, respectively, then k-NN classifier approaches the optimal Bayes classifier (Fukunaga, 1972). One such value of k for which the limiting conditions are satisfied is $\sqrt{T}$).

For MLP different structures were designed for each problem and were used and trained in MATLAB 7.0. These structures were selected experimentally; no optimization technique was used because a traditional MLP classifier was considered for comparing the results.

For GA-classifier a fixed population size of 20 was chosen. The crossover probability was fixed at 0.8 and a variable value of mutation probability was selected from the range [0.015, 0.333]. Initially was assumed a high value, gradually decreasing at first, and then increasing again in the later stages of the GA (Bandyopadhyay et al., 1999).

Also for PS-classifier, a swarm size of 20 was selected. A large initial inertia weight equal to 0.7 was selected to facilitate global search and then decreasing it lower than 0.2 for local search.

AN-classifier was constructed by nine first layer units and three second-layer units. Each first layer units has a number of automata just equal to the dimensions of the feature vectors. Each automaton had nine actions which were $\{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$ and uniform conditions were used (Sastry and Thathachar, 1999; Thathachar and Sastry, 2002).

Experimental results

The proposed LA-classifier is tested on the data sets described in Section 4.1. To estimate accurate generalization rates for the proposed classifier, 2-fold cross validation (*2CV*), 4-fold cross validation (*4CV*), and 10-fold cross validation (*10CV*) are used. It means that 50% (for *2CV*), 25% (for *4CV*), and 10% (for *10CV*) of whole training samples are randomly considered as testing points (validation sets) and others as traditional training set for adjusting model parameters in the classifier. The validation sets is used to estimate the generalization of classifier. Generally, for *m-fold cross validation* (here $m = 2$, 4, and 10) the whole training set is randomly divided into m disjoint sets of equal size. Then the classifier is trained m times, each time with a different set held out as a validation. The estimated performance is the mean of these m score of recognition.

[1] This data set is available at University of California, Irvine, via anonymous ftp ftp.ics.uci.edu/pub/machine-learning-databases.

[2] This data set is available at http://www.ics.uci.edu/~mlearn/MLRepository.html.

Tables 1–5 present the results corresponding to Iris, Wine, Glass, Appendicitis, and Cancer data classification for *2CV*, *4CV*, and *10CV*. The number of hyperplanes in LA-classifier was obtained manually for each problem. Since LA-classifier has a randomized nature in its search process, the results of this classifier obtained by averaging over ten times repetitions.

It is shown in Table 1 that for *4CV* and *10CV* the recognition ability of the classifier is considerably high for three classes for Iris data, which is known to have a very small overlap.

Table 2 shows the classwise and overall recognition scores for Wine data, which are again considerably high for *4CV* and *10CV*.

Table 3 presents the results on all six classes of Glass data. Although the obtained scores of recognition for all *cross validations* are lower than other problems, but it should be mentioned that the Glass data is the overlapping and nonlinear benchmark in pattern recognition. These results conform to earlier findings, when these set of data was used for classifier evaluation in the literature.

Tables 4 and 5 show the performances of the algorithm on the Appendicitis and Cancer data respectively. The overall recognition scores for these two data sets (like Iris, Wine and Glass data sets) show a gradual decrease with decreasing the value of the number of training points. Considerable performances of 92.8% and 94.4% were obtained for Appendicitis and Cancer data at *10CV* respectively.

A comparison of the performance of the LA-classifier is made with that of the k-NN classifier, MLP, GA-classifier, PS-classifier, and AN-classifier for *10CV*. The results are presented in Tables 6–10 for Iris, Wine, Cancer, Glass, and Appendicitis data.

Table 1
Recognition scores (%) for Iris data with $H = 4$ hyperplanes

	2CV	*4CV*	*10CV*
Class1	87.2	92.9	95.7
Class2	86.1	92.4	94.9
Class3	93.1	93.1	95.1
Overall	88.8	92.8	95.2

Table 2
Recognition scores (%) for Wine data with $H = 5$ hyperplanes

	2CV	*4CV*	*10CV*
Class1	68.1	85.3	95.1
Class2	73.8	88.9	94.2
Class3	76.1	85.6	93.4
Overall	72.7	86.6	94.2

Table 3
Recognition scores (%) for Glass data with $H = 6$ hyperplanes

	2CV	*4CV*	*10CV*
Class1	57.1	67.1	73.9
Class2	40.4	60.9	78.3
Class3	50.8	63.5	73.4
Class4	55.8	61.8	83.5
Class5	59.5	68.9	75.2
Class6	68.8	75.5	79.6
Overall	55.4	66.3	77.3

Table 4
Recognition scores (%) for Appendicitis data with $H = 3$ hyperplanes

	2CV	*4CV*	*10CV*
Class1	73.5	85.7	92.6
Class2	80.5	88.0	93.1
Overall	77.0	86.7	92.8

Table 5
Recognition scores (%) for Cancer data with $H = 3$ hyperplanes

	2CV	*4CV*	*10CV*
Class1	85.6	91.2	93.6
Class2	81.6	89.9	95.2
Overall	83.6	90.5	94.4

Table 6
Comparative recognition scores (%) for Iris data with $H = 4$ and *10CV*

	k-NN	MLP	GA-classifier	PS-classifier	AN-classifier	LA-classifier
Class1	94.3	95.4	94.3	92.8	90.5	95.7
Class2	95.2	93.2	96.6	93.5	92.1	94.9
Class3	93.2	93.1	93.9	94.0	93.4	95.1
Overall	94.2	93.9	94.9	93.4	92.0	95.2

Table 7
Comparative recognition scores (%) for Wine data with $H = 5$ and *10CV*

	k-NN	MLP	GA-classifier	PS-classifier	AN-classifier	LA-classifier
Class1	93.5	87.5	93.7	95.6	90.0	95.1
Class2	91.4	83.4	93.1	94.2	87.5	94.2
Class3	90.3	88.3	95.3	92.0	89.3	93.4
Overall	91.7	86.4	94.0	93.9	88.9	94.2

Table 8
Comparative recognition scores (%) for Glass data with $H = 6$ and *10CV*

	k-NN	MLP	GA-classifier	PS-classifier	AN-classifier	LA-classifier
Class1	53.6	56.3	71.3	77.1	66.0	73.9
Class2	67.9	63.7	75.1	78.3	57.1	78.3
Class3	74.2	66.2	78.8	79.2	59.5	73.4
Class4	72.4	55.1	84.8	85.5	63.4	83.5
Class5	73.1	59.9	82.3	78.6	76.8	75.2
Class6	81.4	71.3	75.5	77.0	69.9	79.6
Overall	70.4	62.1	78.0	79.3	65.4	77.3

Table 9
Comparative recognition scores (%) for Appendicitis data with $H = 3$ and *10CV*

	k-NN	MLP	GA-classifier	PS-classifier	AN-classifier	LA-classifier
Class1	89.7	87.3	90.0	89.2	84.9	92.6
Class2	88.8	90.1	94.4	92.7	87.6	93.1
Overall	89.2	88.7	92.2	90.9	86.2	92.8

Table 10
Comparative recognition scores (%) for Cancer data with $H = 3$ and *10CV*

	k-NN	MLP	GA-classifier	PS-classifier	AN-classifier	LA-classifier
Class1	94.1	90.2	94.7	95.1	90.6	93.6
Class2	92.9	89.6	93.1	94.7	89.8	95.2
Overall	93.5	89.9	93.9	94.9	90.2	94.4

For Iris data (Table 6) the proposed classifier (LA-classifier) provides the best overall recognition score. The GA-classifier and k-NN have the nearest performances to the LA-classifier by 0.3% and 1% differences respectively. In fact all the six classifiers appeared in Table 6 have a good performance because the Iris data has not a considerable complexity in the feature space.

For Wine data classification the proposed LA-classifier provides the best overall recognition score. This is followed by the scores for other classifiers in Table 7. The results in Table 7 report 7.8% and 5.3% improvement for the overall recognition score of LA-classifier in comparison to MLP and AN-classifier respectively. Also the LA-classifier gives 2.5% overall recognition score well than the k-NN for Wine data classification. In this case the performances of LA-classifier, GA-classifier, and PS-classifier are similar together.

Table 8 shows that the performances of two novel genetic and swarm intelligence based classifiers (GA-classifier and PS-classifier) are better than the proposed classifier for Glass data classification; but only by differences of 0.7% and 2% respectively. Also Table 8 shows that the LA-classifier has considerable performance differences in comparison to its similar method (AN-classifier) and two conventional classifiers (k-NN and MLP).

Table 9 demonstrates the effectiveness of the LA-classifier, where it has the best performance among all classifiers for Appendicitis data classification.

For Cancer data classification, PS-classifier provides the best performance of 94.9% (Table 10). Table 10 shows only 0.5% difference in performance of the proposed classifier in comparison to the PS-classifier. Amounts of 4.5% and 0.9% are the differences of the overall recognition scores between the proposed LA-classifier and conventional MLP and k-NN classifiers for Cancer data (Table 10). Comparison of the LA-classifier and AN-classifier gives 4.2% improvement for the proposed classifier for Cancer data (Table 10).

Also Tables 6–10 show that the performances of the proposed classifier are comparable to, sometimes better than the GA-classifier and PS-classifier.

An investigation on the standard-deviation of extensive experimental results, reported in Tables 6–10 showed that the standard-deviation of the proposed method likewise GA-classifier, PS-classifier, and k-NN are close together and lower than that of other classifiers for all benchmarks for *10CV*. It means a good robustness for the proposed method, as well as PS-classifier, GA-classifier, and k-NN.

4.3. The effect of the number of training points

To show the effect of the number of training points in the performance of the LA-classifier, comparative results with Bayes classifier are presented. The Bayes classifier is one of the most widely used in statistical pattern recognition which provides optimal performance from the standpoint of error probabilities in a statistical framework. It is known the best classifier when the class distributions and the a priori probabilities are known. Consequently, the desirable property of any classifier is that it should approximate or approach the Bayes classifier under limiting conditions.

Some parametric and non-parametric techniques have been introduced for density function estimation (Tou and Gonzalez, 1992) (e.g. maximum likelihood, Gaussian mixture models, histogram, and Parzen windows). In this article Parzen windows is used to estimate the density function of the benchmarks. Also a priori probabilities of $\frac{T_i}{T}$ for totally T training samples and T_i patterns from class i, are considered.

Table 11 presents the comparative recognition scores (%) for LA-classifier and Bayes classifier for *2CV*, *4CV*, and *10CV*. In this table the numbers of hyperplanes (H) for LA-classifier are 4, 5, 6, 3, and 3 for Iris, Wine, Glass, Appendicitis, and Cancer data respectively. From the

Table 11
Comparative recognition scores (%) for LA-classifier and Bayes classifier for all data sets

	2CV		*4CV*		*10CV*	
	LA-classifier	Bayes classifier	LA-classifier	Bayes classifier	LA-classifier	Bayes classifier
Iris	88.8	92.1	92.8	96.1	95.2	97.5
Wine	72.7	92.2	86.6	94.4	94.2	96.7
Glass	55.4	68.0	66.3	71.3	77.3	78.8
Appendicitis	77.0	86.8	86.7	94.8	92.3	97.3
Cancer	83.6	92.3	90.5	95.7	94.4	98.4

results appeared in Table 11 it is found that as the number of training points increases the performance of the proposed LA-classifier tends to the Bayes classifier for all data sets. This confirms the effectiveness of the LA-classifier in estimation of the hyperplanes which are near to the optimum one for the large number of training points.

Table 11 shows that the largest difference between the performance of LA-classifier and Bayes classifier for *10CV* appears for Appendicitis data classification by 5%. For Iris data, Wine data, and Glass data we see a little difference between the score of recognition for these two classifiers (for *10CV*). Those are 2.3%, 2.5%, and 1.5% for Iris data, Wine data, and Glass data respectively. For Cancer data it is found that the performance of Bayes classifier has better rate of 4% for *10CV*.

In fact the obtained results in Table 11 conform to this proved theorem that "*for a large number of training points*, *the performance of any classifier*, *whose the criterion is to reduce the number of misclassified points*, *approaches that of the Bayes classifier*" (Bandyopadhyay et al., 1999).

5. Discussion and conclusion

In this paper, a pattern-classification methodology based on learning automata is proposed (called LA-classifier). Multi-rate cross-validation methodology and different kinds of benchmarks (with nonlinear, overlapping class boundaries and dimensions ranging from four to thirteen) provide extensive experimental results that indicate for a given value of the number of hyperplanes (H), the LA-classifier can effectively approximate the weight vectors of decision functions to separate reference classes in feature space. The performance of the classifier is also found to be comparable to, sometimes better than, those of the conventional (k-NN and MLP), evolutionary (GA-classifier), and swarm intelligence based (PS-classifier) classifiers. Also the performance of the LA-classifier is obtained more effective than AN-classifier which is a three-layer network consisting of teams of automata for pattern classification. Finally it is found that as the number of training points increases the performance of LA-classifier tends to the performance of Bayes classifier which is an optimal conventional classifier (but with some limitations in usage due to need to important *priori* knowledge).

Regarding the timing requirements, it may be noted that the LA-classifier takes a large amount of time *during training* like MLP, GA-classifier, PS-classifier and AN-classifier; because it has been established on the function optimization and like the other optimization algorithms it needs a large amount of time for convergence. However, the time taken *during testing is very small* for the proposed classifier; because the decision hyperplanes have been already obtained in training phase. On the contrary, in high dimensional feature spaces the k-NN classifier takes significant amount of time for testing.

A theoretical analysis of the LA-classifier, study on over-fitting, over-learning, reliability, and resemblance of the LA-classifier and study on how its parameters affect convergence and performance are topic tasks for future works.

References

Bandyopadhyay, S., Murthy, C.A., Pal, S.A., 1999. Theoretical performance of genetic pattern classifier. Int. J. Franklin Inst. 336, 387–422.

Beygi, H., Meybodi, M.R., 2006. A new action-set learning automaton for function optimization. Int. J. Franklin Inst. 343, 27–47.

Fukunaga, K., 1972. Introduction to Statistical Pattern Recognition. Academic, New York.

Obaidat, M.S., Papadimitriou, G.I., Pomportsis, A.S., 2003. Efficient fast learning automata. Inform. Sci. 157, 121–133.

Oommen, B.J., St.Criox, E.V.de, 1996. Graph partitioning using learning automata. IEEE Trans. Comput. 45, 195–208.

Oommen, B.J., Lanctôt, J.K., 1990. Discretized pursuit learning automata. IEEE Trans. Systems Man Cybernet. 20, 931–938.

Sastry, P.S., Thathachar, M.A.L., 1999. Learning automata for algorithm for pattern classification. Sadhana 24, 261–292.

Shapiro, I.J., Narendra, K.S., 1969. Use of stochastic automata for parameter self-optimization with multi-modal performance criteria. IEEE Trans. Systems. Man Cybernet. 14, 323–334.

Tang, C.K.K., Mars, P., 1993. Games of stochastic learning automata and adaptive signal processing. IEEE Trans. Systems. Man Cyberet. 23, 851–856.

Thathachar, M.A.L., Sastry, P.S., 1985. A class of rapidly converging algorithms for learning automata. IEEE Trans. Systems. Man Cybernet. 15, 168–175.

Thathachar, M.A.L., Sastry, P.S., 2002. Varieties of learning automata: An overview. IEEE Trans. Systems. Man Cybernet. Part B: Cybernet. 32, 711–722.

Tou, J.T., Gonzalez, R.C., 1992. Pattern Recognition Principles. APMCC.

Wu, Q.H., 1995. Learning coordinated control of power systems using inter-connected learning automata. Int. J. Electr. Power Energy Syst. 17, 91–99.

Zahiri, S.-H., Seyedin, S.-A., 2007. Swarm intelligence based classifiers. Int. J. Franklin Inst. 344, 362–376.

Zeng, X., Liu, Z., 2005. A learning automaton based algorithm for optimization of continuous complex function. Inform. Sci. 174, 165–175.

Zeng, X., Zhou, J., Vasseur, C., 2000. A strategy for controlling non-linear systems using a learning automaton. Automatica 36, 1517–1524.