Available online at www.sciencedirect.com
ScienceDirect
Journal of the Franklin Institute 344 (2007) 362–376
Journal of The Franklin Institute
www.elsevier.com/locate/jfranklin

Swarm intelligence based classifiers ☆

Seyed-Hamid Zahiri[a,*], Seyed-Alireza Seyedin[a,b]

[a]*Department of Electrical Engineering, Faculty of Engineering, Birjand University, P.O. Box. 97175-376, Birjand, Iran*
[b]*Department of Electrical Engineering, Faculty of Engineering, Ferdowsi University of Masshad, Mashhad, Iran*

Received 14 January 2005; accepted 21 December 2005

Abstract

A proposed *particle swarm classifier* has been integrated with the concept of *intelligently controlling the search process of PSO* to develop an efficient swarm intelligence based classifier, which is called intelligent particle swarm classifier (IPS-classifier). This classifier is described to find the decision hyperplanes to classify patterns of different classes in the feature space. An intelligent fuzzy controller is designed to improve the performance and efficiency of the proposed classifier by adapting three important parameters of PSO (*inertia weight*, *cognitive parameter* and *social parameter*). Three pattern recognition problems with different feature vector dimensions are used to demonstrate the effectiveness of the introduced classifier: Iris data classification, Wine data classification and radar targets classification from backscattered signals. The experimental results show that the performance of the IPS-classifier is comparable to or better than the *k*-nearest neighbor (*k*-NN) and multi-layer perceptron (MLP) classifiers, which are two conventional classifiers.

Keywords: Decision hyperplanes; Fuzzy controller; Particle swarm optimization; Pattern recognition

1. Introduction

Particle swarm optimization (PSO) is a swarm intelligence technique developed by Kennedy and Eberhart in 1995 [1]. In fact, natural flocking and swarm behavior of birds

☆This paper is an extended version of the paper entitled "Intelligent Particle Swarm Classifiers", which has been presented in the *First International Conference on Modeling, Simulation and Applied Optimization* (*ICMSAO/05*) and has been selected for the special issue of Journal of the Franklin Institute.

*Corresponding author. Tel.: +985612227044; fax: +985612223925.
E-mail addresses: shzahiri@yahoo.com (S.-H. Zahiri), seyedin@ferdowsi.um.ac.ir (S.-A. Seyedin).

0016-0032/$30.00
doi:10.1016/j.jfranklin.2005.12.006

and insects inspired him to PSO. This technique has been used in several optimization and engineering problems (e.g. [2,3]). In pattern recognition problems some particle swarm clustering techniques have been proposed (e.g. [4,5]), but a swarm intelligence based classifier using PSO directly to obtain the decision functions in the feature space has not been implemented in the recent researches. In this paper an *intelligent particle swarm classifier* (*IPS-classifier*) is developed, integrating the concept of *intelligently controlling the search process of* PSO with a proposed *particle swarm classifier* (*PS-classifier*), which is designed to find the decision hyperplanes in the feature space.

An IPS-classifier has an additional intelligent controller to adapt the important parameters of PSO to increase its efficiency and means steering the swarm to an appropriate trajectory to find a better solution. Simulation results indicate that convergence to better hyperplanes with a lower number of iterations can be achieved using this technique. In fact, the IPS-classifier searches the solution space and chooses hyperplanes in such manner that the misclassified training points are minimized.

PSO is a simple and powerful search technique in high dimensional spaces, therefore IPS-classifier has the potentiality to successfully classify different classes in *high dimensional* feature spaces by achieving the separate hyperplanes, with a little priori information.

Any intelligent controller may be utilized to increase the efficiency of the classifier. In this article a fuzzy structure has been chosen and the IPS-classifier using this controller is called *fuzzy controlled particle swarm classifier* (*FCPS-classifier*).

The fuzzy controller rules have been extracted from a linguistic description from previous researches on study the effects of the PSO parameters on its search process (e.g. [6–8]).

Two common benchmark problems and a special problem in pattern recognition are considered to compare the proposed and the other methods with each other. The Iris data and the Wine data classification are common problems in pattern recognition researches with low and medium feature space dimensions and automatic target recognition for continuous wave radars is a special pattern recognition problem with high feature space dimensions. The performance of an IPS-classifier is compared with k-NN and MLP classifiers to show that the average recognition scores of designed IPS-classifier are better than or comparable to the traditional classifiers. To show the effectiveness of the intelligent fuzzy controller in the search process and correctly steering the swarm toward the solution, some meaningful figures are included.

In this paper, Section 2 explains a particle swarm classifier (PS-classifier). Intelligent particle swarm classifier is described in the next Section. Section 4 considers implementation of the classifier and experimental results on the three aforesaid pattern recognition problems. Finally, conclusion and discussion is presented in Section 5.

2. Fundamentals of a particle swarm classifier

2.1. PSO algorithm

In the basic PSO technique proposed by Kennedy and Eberhart [1], great number of particles moves around in a multi-dimensional space and each particle memorizes its position vector and velocity vector as well as the time at which the particle has acquired the best fitness. Furthermore, related particles can share data at the best-fitness time.

The velocity of each particle is updated with the best positions acquired for all particles over iterations and the best positions are acquired by the related particles over generations.

To improve the performance of the basic PSO, some new versions of it have been proposed. At first, the concept of an *inertia weight* was developed to better control exploration and exploitation in [9]. Then, the research done by Clerc [10] indicated that using a *constriction factor* may be necessary to insure convergence of the particle swarm algorithm. After these two important modifications in the basic PSO, the *multi-phase particle swarm optimization* (*MPSO*), the *particle swarm optimization with Gaussian mutation*, the *quantum particle swarm optimization*, a modified PSO with *increasing inertia weight schedule*, the *Gaussian particle swarm optimization* (*GPSO*) and the *guaranteed convergence PSO* (*GCPSO*) were introduced in [11–16], respectively.

In this paper the PSO with the inertia weight is used; because a good knowledge about the influence of existing parameters in this version of PSO on its search process is available. "*Gbest*" strategy is considered in this paper, as well. It means that particles have the information of the entire swarm rather than only their own and their neighbors' bests, which is called "*lbest*". Conceptually, *gbest* connects all the particles together, which means that their social interaction is maximal. In contrast, *lbest* results in a local neighborhood for the particle. Increasing the neighborhood size to the entire swarm increases the convergence rate. However, it might be trapped into a local optimum. Since, we design an additional fuzzy controller to steer the swarm toward a better solution and escape it from local traps, it adds the computational cost compared to a simple PS-classifier, thus *gbest* is chosen instead of *lbest* for its higher convergence rate.

In this approach updating is executed by the following equations:

$$V_i^{q+1} = \omega.V_i^q + c_1\phi_1(P_i - Y_i) + c_2\phi_2(P_g - Y_i), \tag{1}$$

$$Y_i^{q+1} = Y_i^q + V_i^q. \tag{2}$$

In these relations $i = 1, 2, \ldots, Pop$, where "*Pop*" is the swarm size, q is the generation counter, $V_i = (v_{i1}, v_{i2}, \ldots, v_{in})$ is the velocity vector of the ith particle of the swarm, $P_i = (p_{i1}, p_{i2}, \ldots, p_{in})$ denotes the best position that has been visited by the particle, $Y_i = (y_{i1}, y_{i2}, \ldots, y_{in})$ is the current position and $P_g = (p_{g1}, p_{g2}, \ldots, p_{gn})$ is the best particle among the particles in the swarm. ϕ_1 and ϕ_2 are random numbers uniformly distributed in the range (0,1) and n is the dimension of the space. c_1 and c_2 are positive constants, called *cognitive* and *social* parameters, respectively, both equal to 2 in general cases. ω is called inertia weight and is brought into PSO to balance the global and local search ability. A large inertia weight facilitates a global search, while a small inertia weight facilitates a local search [17,18].

In spite of PSO with constriction coefficient that requires no explicit limitation as upper bound $V_{\max}$, in aforesaid case of PSO an upper and lower bounds, $V_{\max}$ and $V_{\min}$, are used to constrain the velocities of particles. Very small values of speed will cause the particles to get trapped in local optima. On the other hand, very large values can cause the particles to oscillate around a point [19].

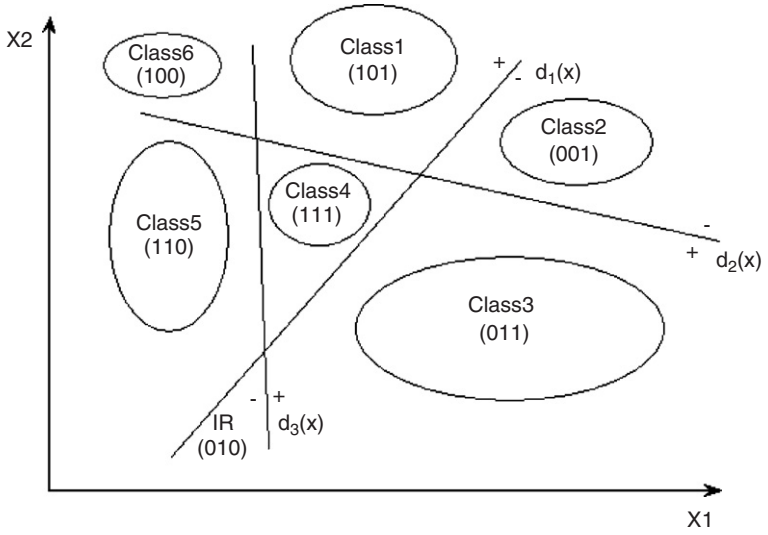

Fig. 1. Each region can identify an individual class by its code obtained from the signs of hyperplanes.

2.2. Particle swarm classifier

A *particle swarm classifier* (*PS-classifier*) has three major parts including decision hyperplanes, fitness function definition, and structure.

2.2.1. Decision hyperplanes

A general hyperplane is in the form of

$$\mathrm{d}(X) = w_1x_1 + w_2x_2 + \cdots + w_nx_n + w_{n+1}, \tag{3}$$

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, that each region distinguishes an individual class (Fig. 1). In Fig. 1 each class belongs to a region encoded by the sign of three hyperplanes (for two-dimensional feature space). In this figure, IR denotes the indeterminate region. Some especial cases are described in [20].

The *PS-classifier* must find $W_j, (j = 1, 2, \ldots, H)$ in solution space, where H is the necessary number of decision hyperplanes.

2.2.2. Fitness function definition

Fitness function is defined as follows:

$$fit(P_i) = T - Miss(P_i), \tag{4}$$

where T is the size of the total training data set and $Miss(P_i)$ is the number of misclassified training points by P_i. By maximizing the fitness function, minimum error of training points classification is achieved.

2.2.3. The structure of particle swarm classifier

According to the above descriptions, designing a PS-classifier has the following steps:

Swarm = *Generate*(*Swarm_size*);
$q = 1$;
$c_1 = 2$;
$c_2 = 2$;
$\omega = 0.9$;
while (*termination condition*)
for $i = 1$ *to Pop*
Compute_fitness(Y_i);
  P_g = *obtain* (P_g);
  P_i = *Obtain* (P_i);
  Create(ϕ_1, ϕ_2);
  $V_i^q = \omega.V_i^{q-1} + c_1\phi_1(P_i - Y_i) + c_2\phi_2(P_g - Y_i)$;
    if $V_i(q) > V_{\max}$
      $V_i(q) = V_{\max}$;
    else if $V_i(q) < V_{\min}$
      $V_i(q) = V_{\min}$;
    end if;
  $Y_i^{q+1} = Y_i^q + V_i^q$;
  end for;
Linear_decrease(ω);
  $q = q + 1$;
  end while;

In a PS-classifier each particle is selected randomly from the solution space and has the form of $P = [W_1, W_2, \ldots, W_i, \ldots, H]'$ where $W_i = (w_{i1}, w_{i2}, \ldots, w_{in}, w_{in+1})$ is the weight vector of ith hyperplane and H is the pre-defined number of hyperplanes. Fitness function is defined as Eq. (5). Termination condition can be the best fitness value or a default maximum number of iterations. For an effective search in the solution space, we chose a linearly decreasing schedule for inertia weight (ω), as it has been mentioned in [17] and [18] for the local and global search tuning. $V_{\max}$ is limited to $Y_{\max}$ and $V_{\min}$ is considered equal to $Y_{\min}$.

After an enough number of iterations, the particles converge to a solution that is the decision hyperplanes whose misclassified training points are minimized.

3. Intelligent particle swarm classifier

It is known that the parameters of PSO play major roles in its search process (e.g. premature convergence, convergence rate, local capturing, exploitation, exploration, etc.). In the PSO defined in Section 2.1 *inertia weight*, *cognitive parameter* and *social parameter* are these three important factors. In the most of reported researches on the applications of PSO, these parameters have been achieved by running the PSO for several times with different sets of parameter to find a proper set. On the other hand, the search process of the

PS-classifier is nonlinear and very complicated and it is hard—if not impossible—to model mathematically the search process to adjust adaptively the PS-classifier parameters. But over the years, some understanding of the PSO search process has become accumulated and linguistic description of the search process is currently available. New knowledge makes the fuzzy system a good candidate for controlling the aforementioned PS-classifier parameters, intelligently.

It should be mentioned that the efficiency and complexity in the most of evolutionary classifiers have direct relationship with each other. The more efficiency you need the more complexity your system should have and vice versa. Similarly, although using an additional intelligent controller (here fuzzy controller) in a PS-classifier increases the computation complexity, it improves the efficiency of the PS-classifier, especially in high dimensional feature space.

Already, in [21] a fuzzy system has been designed to adjust the inertia weight of PSO for the Rosenbrock function and in [6] this fuzzy system has been modified to fit a wide range of optimization problems.

In this paper we introduce another fuzzy system to control the inertia weight, cognitive parameter and social parameter to improve the efficiency and performance of the proposed PS-classifier. To extract some effective fuzzy rules, at first, a linguistic description about the effects of the PSO parameters on its search process is presented as a subsection.

3.1. Linguistic description on the effect of PSO parameters on its search process

3.1.1. Inertia weight (ω)

The role of the inertia weight ω is very important in PSO convergence behavior. The inertia weight is employed to control the effect of the previous velocities on the current velocity. In this way, the parameter ω makes a compromise between the global (wide-ranging) and local (nearby) exploration abilities of the swarm. A large inertia weight facilitates global exploration (searching new areas), while a small one tends to facilitate local exploration.

A suitable value of the inertia ω usually provides balance between global and local exploration abilities and consequently a more efficient search to locate the optimum solution can be made. A general rule of thumb suggests that it is better to initially set the inertia to a large value, in order to make better global exploration of the search space, and gradually *decrease* it to get more refined solutions.

In fact the PSO search process is a nonlinear and complicated process and linearly *decreasing* or *increasing* inertia weight changes the search from global to local and local to global linearly, as well. However, there are many problems that require the search algorithm to have nonlinear search ability. By deriving some statistical features from the obtained results in each iteration, may help to understand the quality of the PSO search and to calculate a proper inertia weight for the next iteration.

3.1.2. Cognitive parameter (c_1) and social parameter (c_2)

In the PSO algorithm, each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. The fitness value is also stored and is called *pbest*. Another "*best*" value tracked by the global version of the PSO is the overall best value and its location obtained so far by any particle in the swarm (i.e. *gbest*). The particle swarm optimization concept, at each step, consists of changing the

velocity of each particle (accelerating) toward its *pbest* and *gbest* locations. Acceleration is weighted by a random term with separate random numbers generated for acceleration toward *pbest* and *gbest* locations. The acceleration constants c_1 and c_2 represent the weighting of the stochastic acceleration terms that pull each particle toward *pbest* and *gbest* positions. Low values allow particles to roam far from target regions before being tugged back, while high values result in abrupt movement toward target regions. Early experience with PSO (mostly trail and error) led us to set the acceleration constants c_1 and c_2 equal to 2.0, but it is not a usual rule.

In the search process, sometimes *social* behavior may be more important than *cognitive* behavior and vice versa. Regarding some statistical parameters, which are calculated in each iteration, it is possible to tune the *cognitive parameter* and *social parameter* adaptively, to steer the whole swarm and each particle toward proper trajectory. For example, in the beginning of the search process the global search is more important than local search. By tuning the inertia weight and the social parameter to a large value and setting the cognitive parameter to a small value, the global search ability and maximal interaction in the swarm is achieved. Contrariwise, after enough iteration, when a good position (with a good fitness) is achieved, the cognitive parameter must be increased and social parameter besides the inertia weight must be decreased to emphasize local search ability. As the fitness value of a particle becomes better and better, part of search space explored by the particle should become smaller and smaller. It means that social interaction should be decreased. On the other hand, less improvement in the particle fitness causes a sociality search for the exploration. This means that the value of the cognitive parameter should be decreased and the value of the social parameter should be increased.

3.2. The fuzzy controlled PS-classifier (FCPS-classifier)

The fuzzy controlled PS-classifier (FCPS-classifier) is a kind of IPS-classifiers, whose controller is a fuzzy structure with some fuzzy *inputs*, *outputs* and *IF ... THEN ... fuzzy rules* were extracted from the linguistic description presented in Section 3.1.

The fuzzy controller is constructed with three inputs and three outputs. The inputs are as follows:

- f_{best}: the maximum fitness value that is achieved in qth iteration in whole the swam;
- *P-dist*: the norm of difference between P_i and P_g for ith particle;
- *UN*: the number of iterations, in those the best fitness value of the swarm remain unchanged.

The three outputs are

- ω_i: the inertia weight for ith particle;
- c_{1i}: the cognitive parameter of the ith particle;
- c_{2i}: the social parameter of the ith particle.

f_{best} indicates the value of evolution of the swarm. High values of f_{best} means that at least one particle has been reached to a suitable position and other particles must steer toward it. Low values of f_{best} means a deviation from the proper trajectory, thus the local search should be emphasized instead of global search.

UN indicates that the basic search loop of PS-classifier has been repeated for several iterations without any improvement of the best-achieved fitness value. It means that PS-classifier is probably captured in some nonimportant local solutions and the global search ability should become stronger than local search ability by a proper tuning the inertia weight, cognitive and social parameters.

Since P_i includes the best positions of each particle and P_g is the same for the *swarm*, *P-dist* can be a metric for evaluation the population diversity in a PS-classifier. Large value of *P-dist* means that in the last iterations the diversity of swarm was high and vice versa. Both P_i and P_g are matrices, which have the form: $P_i = [W_{1i}, W_{2i}, \ldots, W_{Hi}]'$ and $P_g = [W_{1g}, W_{2g}, \ldots, W_{ig}, \ldots, W_{Hg}]'$, respectively. *P-dist* is defined as follows:

$$P\text{-}dist = \|P_g - P_i\| = \sum_{k=1}^{H} |W_{kg} - W_{ki}|, \tag{5}$$

where H is the number of hyperplanes , W_k denotes a hyperplane as a row of both P_i and P_g and $|\cdot|$ means the norm of the $(W_{kg} - W_{ki})$ vector.

According to above descriptions, fuzzy combination of f_{best}, *P-dist* and *UN* finds a proper approximation of some important information of the PS-classifier search process (e.g. local capturing, exploration, exploitation, global search, local search and etc.). For example if f_{best} is low*AND P-dist* is low too, it means that the swarm is caught in an unsuitable location in the solution space. In this case it is reasonable to increase the inertia weight to reach the large values of velocity and scattering the particles in a wide region in the solution space. At same time the exploration must be emphasized. This can be implemented by decreasing and increasing the social and cognitive parameters, respectively.

If the basic loop of PS-classifier is repeated for several iterations without any improvement of the best-achieved fitness value (*UN* takes large values, while f_{best} has a medium value), it means that PS-classifier is probably captured in some local solutions and above behavior of intelligent controller must be repeated.

Contrariwise, high fitness values achievement (when f_{best} is high), while *UN* has a large value means approaching to the final solution. Thus, the inertia weight and cognitive parameter should be decreased and social parameter must be increased.

In the same way six fuzzy rules are extracted regarding the values of defined fuzzy inputs and using the descriptions appeared in Section 3.1.

(1) IF f_{best} is low and *P-dist* is low THEN ω_i is high, c_{1i} is high, and c_{2i} is low.
(2) IF f_{best} is medium and *UN* is high and *P-dist* is high THEN ω_i is medium, c_{1i} is high, and c_{2i} is medium.
(3) IF f_{best} is medium and *UN* is low and *P-dist* is high THEN ω_i is medium, c_{1i} is medium, and c_{2i} is medium.
(4) IF f_{best} is high and *UN* is high THEN ω_i is low, c_{1i} is low, and c_{2i} is high.
(5) IF f_{best} is low and *UN* is high THEN ω_i is high, c_{1i} is low, and c_{2i} is high.
(6) IF f_{best} is high and *P-dist* is high THEN ω_i is low, c_{1i} is low, and c_{2i} is high.

The fuzzy controller has been designed with above fuzzy rules and inputs and outputs membership functions, which are shown in Fig. 2. Since the fuzzy inputs values change from one problem to another, the scaled values of fuzzy inputs have been shown in Fig. 2.

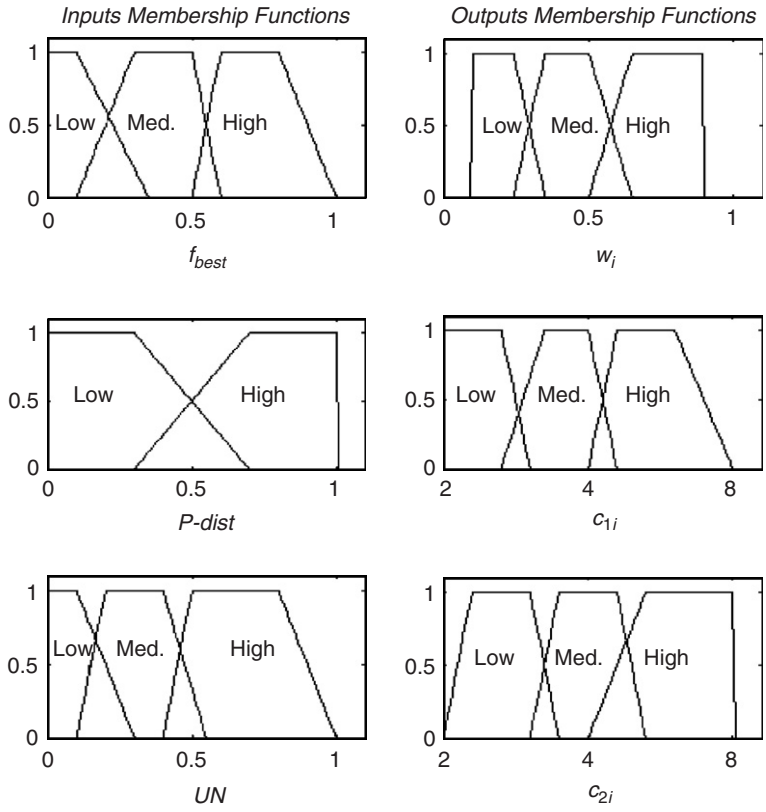

Fig. 2. Inputs and outputs membership functions.

Contrariwise, the output membership functions are appeared in this figure by their practical values. We used *Mamdani* fuzzy model with *centroid defuzzification* way, and Min and Max for *T-norm* and *S-norm* operators, respectively.

It should be mentioned that different kinds of inputs, outputs, membership function shapes, membership function locations and fuzzy rules may be introduced and even these parameters can be optimized by another optimization algorithm. In this paper the membership functions and their locations are selected and tuned manually.

4. Implementation and experimental results

Three pattern recognition problems with different augmented feature vectors dimensions (5,14,129) are used to show the performance of the IPS-classifier. A description of the data sets is given here.

4.1. Data sets

Iris data: The Iris data contains 50 measurements of four features from each three species Iris setosa, Iris versicolor, and Iris virginica [22]. Features are sepal length, sepal width, petal length and petal width.

Table 1
Ten targets as reference classes

Number	Target	Application
1	V.F3	Training
2	PC-7	Training
3	ANTONOV AN-12	Military
4	FFA AS ZZO118A	Training
5	BAE-248 SERIES 2B	Transportation
6	KJ 500-3S	Military
7	ROLLS ROYCE ALISON	Military
8	KUZNETSORNK-8-2	Transportation
9	TUMMANSKY R-11 F2S	Military
10	ROLLS ROYCE 535 E1 H4	Military

Wine data: The wine data contains the chemical analysis of wines grown in the same region in Italy but derived from different cultivars [23]. The 13 continuous attributes are available for classification. The number of classes is three and the numbers of instances in each class are 59, 71 and 48, respectively.

Radar targets: An application of pattern recognition is automatic target recognition (ATR) for continuous wave radars. In this paper jet engine modulations approach (JEM) is used for this purpose. In this way the modulation of the radar wave, which is caused by rotating propellers and jet engine blades of targets is considered [24]. Ten different flying objects were chosen as introduced in Table 1 for classification in 20° elevation angle [25]. Backscattered signals were simulated using a theoretical model proposed in [26]. After sampling from backscattered signals and data reduction preprocess, we took 128 points FFT as feature vectors for each target.

4.2. Comparison with existing methods

The performance of proposed IPS-classifier is compared with the performance of MLP and k-NN classifiers to show that the average of recognition scores of designed PS-classifier are better than or comparable to the traditional classifiers. For MLP, the [3–5] structure for Iris data, [8,5,4] for Wine data and [21,19,12,14,8,11] for radar targets are used and trained in MATLAB 7.0. These structures were selected experimentally. No optimization technique is used and a *traditional* MLP classifier is considered to compare the results. For these three structures the learning rate η is initially fixed at 2.0. This is decreased by a factor of 2, up to a pre-specified minimum value, if the mean squared error starts oscillating. In case the error decreases very slowly, then the learning rate is doubled. The reason is that most likely the algorithm has confronted a plateau in the error surface.

k-NN classifier is executed taking k equal to $\sqrt{T}$, where T is the total number of training samples (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 [20]. One value of k for which the limiting conditions are satisfied is $\sqrt{T}$).

4.3. Experimental results

The proposed IPS-classifier (i.e. FCPS-classifier), MLP and k-NN classifiers are tested on the data sets described in Section 4.1. Tables 2 and 3 present the results for Iris data and

Table 2
Recognition scores (%) for Iris test data classification for $H = 3$

	Total training points = 15			Total training points = 30			Total training points = 45		
	IPS-classifier	MLP	k-NN	IPS-classifier	MLP	k-NN	IPS-classifier	MLP	k-NN
Class1	75.5	70.2	73.9	81.4	78.3	86.3	90.3	83.4	91.4
Class2	80.8	75.2	80.3	93.3	82.5	90.2	92.4	87.6	93.8
Class3	74.1	71.1	78.7	88.6	81.1	91.4	91.3	85.7	92.0
Overall	76.80	72.17	77.63	87.76	80.63	89.30	91.33	88.7	92.40

Table 3
Recognition scores (%) for Wine test data classification for $H = 5$

	Total training points = 15			Total training points = 30			Total training points = 45		
	IPS-classifier	MLP	k-NN	IPS-classifier	MLP	k-NN	IPS-classifier	MLP	k-NN
Class1	84.3	85.3	88.5	91.1	92.2	92.0	97.1	94.5	97.5
Class2	85.4	81.6	85.1	95.5	93.3	93.8	94.7	95.2	96.5
Class3	80.3	79.0	82.1	86.3	86.8	90.1	94.1	91.1	92.0
Overall	83.33	81.97	85.23	90.97	90.77	91.97	95.30	93.60	95.33

Wine data classifications, respectively. Some different number of training samples ($t = 5, 10, 15$) have been considered. The training samples were chosen randomly from each class. These experiments have been done using three hyperplanes for Iris ($H = 3$) and five hyperplanes for Wine ($H = 5$). The total number of training points is $t *$ *Number of Classes* (e.g. for Iris data for $t = 5$ the total number of training points is $T = 15$). The remained patterns in each case were considered as testing points. The reported results in Tables 2 and 3 are for testing patterns classification. The last rows of these Tables show the overall recognition scores.

Since the IPS-classifier start with a random initial condition and it is possible to converge to different separating hyperplanes the presented results for IPS-classifier are the average on 10 different realizations. For each problem the swarm size was set to 20 and termination condition is considered as a maximum value of iterations, which was set to an experimentally selected value of 250, but if a particle can classify all the training points, the basic loop of IPS-classifier will be terminated. Tables 2 and 3 show that as the number of training points increases, the performance of the IPS-classifier becomes better and better. It is comparable to or better than MLP and k-NN classifiers. For example, in case the total number of training points is equal to 30, for Iris data classification, the recognition score of class2 for IPS-classifier is 95.5%, which is better than both MLP and k-NN. For 30 total training points, this value for Wine data classification is 93.3% which is 10.8% better than MLP and 3.1% better than k-NN results.

Although in some cases it can be seen that the recognition scores for MLP or k-NN is better than the IPS-classifier, the differences between them are not significant. For example the maximum difference, for Iris data classification is equal to 4.2% in comparison to k-NN classifier (for 15 total number of training data points for class1); but the difference

Table 4
Average recognition scores (%) with respect to SNR for radar targets classification for $H = 10$

SNR (dB)	−15	−10	−5	0	5	10	15
IPS-classifier	10.1	14.7	25.4	52.1	61.1	89.2	90.1
MLP	10.3	11.7	14.9	46.5	57.9	66.6	76.2
k-NN	10.5	7.7	17.0	50.5	62.5	79.0	79.5

between overall recognition scores for this number of training points is only 1.9%. For more training points the difference is negligible.

These values demonstrate the ability of IPS-classifier, as a new swarm intelligence based classifier, in comparison to two traditional classifiers MLP and k-NN.

Radar targets classification is done by 10 hyperplanes ($H = 10$) and for 10 numbers of training points from each target. In this experiment we waited for 300 iterations running for IPS-classifier in different signal-to-noise ratios (SNRs) (changing the variance of Gaussian noise produces different noise power). Ten noisy samples from each class for each SNR were considered as the testing points. Table 4 shows the average scores of recognition.

As the SNR increases, the performance of the IPS-classifier considerably improves in comparison to the other classifiers. At SNR = 10 dB and SNR = 15 dB the performance of IPS-classifier is better than MLP, by 22.6% and 13.9%, respectively. Also at the same SNRs, the performance of the IPS-classifier is better than k-NN, by a considerable difference (10.2% at SNR = 10 dB and 10.6% at SNR = 15 dB). Of course there exists only one case in Table 4, at SNR = 5 dB, that the performance of k-NN is better than IPS-classifier, even in this case the difference is not large and is limited to 1.4%.

4.4. Effectiveness of the fuzzy controller

Reduction of the iterations number while running the IPS-classifier expresses a meaningful concept that the fuzzy controller steers particles to converge toward the solution and escaping them from undesired local points (the number of fitness function evaluations can be obtained by: $Swarm\_size * Number\ of\ iterations$).

In some experiments, a simple PS-classifier (which has not any intelligent controller to adapt *the inertia weight*, *cognitive parameter*, *and social parameter*) were designed and its performances on three aforesaid classification problems were evaluated for different number of iterations. The same experiments were executed for an IPS-classifier (here FCPS-classifier), as well. These experiments were repeated 10 times and then their average scores for test points recognition were compared with each other. In these experiments, the number of training data from each class is 10. The remaining points were considered as test data. The swarm size for both the simple PS-classifier and the FCPS-classifier was set to 20 and the cognitive and social parameters in simple PS-classifier are 2. The maximum value of iterations is the same as the FCPS-classifier (250 iterations), but if a particle can classify all the training points the, the basic loop of the IPS-classifier will be terminated.

Fig. 3 demonstrates the fact that the designed fuzzy controller helps the PS-classifier to find a proper trajectory for converging to the solutions with lower iterations number. This improvement is more significant at high recognition scores, because at the first iterations the designed intelligent controller has not enough time to play its important role of

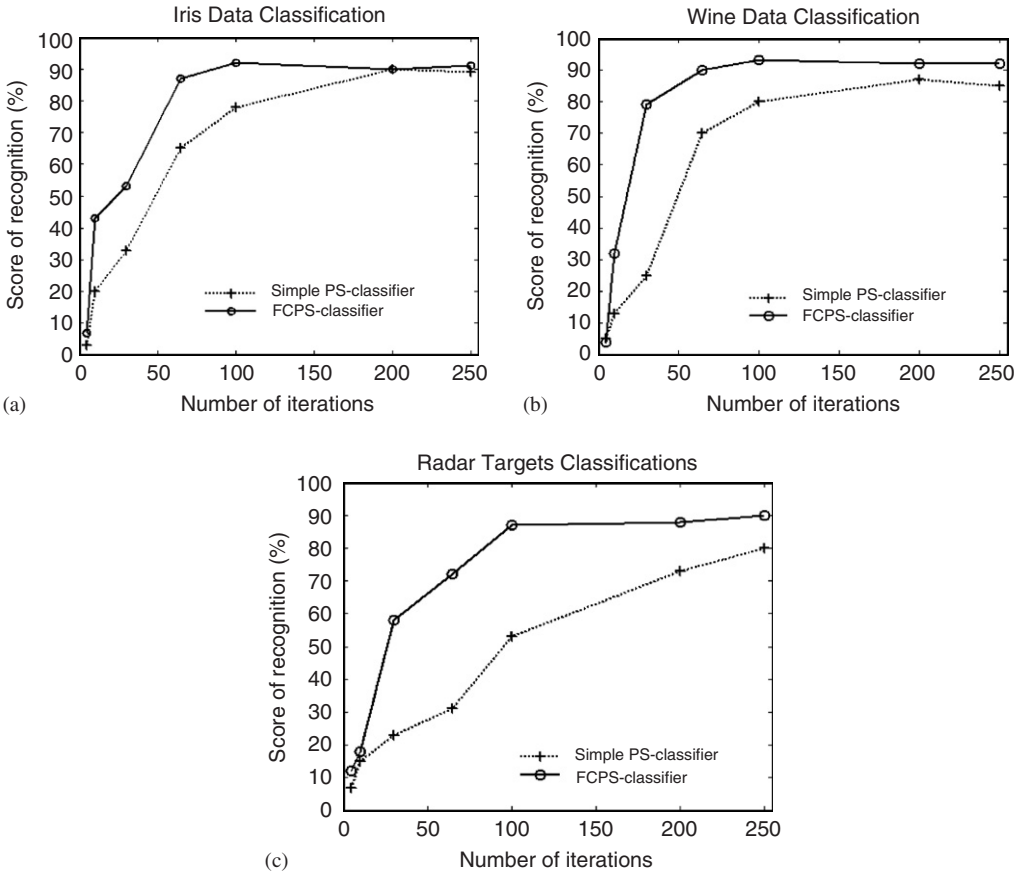

Fig. 3. The average scores of recognition (%) with respect to the number of iterations for (a) Iris data, (b) Wine data, and (c) Radar targets classification.

parameter adaptation in IPS-classifier. Reduction of the iteration number at 90% average recognition score is about 120 (60%) for Iris data classification. This improvement is approximately 67 iterations (33%) at 80% recognition score for Wine data classification and 163 iterations (65.2%) reduction for 80% average score of recognition for radar targets classification (in this step the SNR is approximately 10 dB).

The effectiveness of fuzzy controller is more apparent in feature spaces with more dimensions. The difference between the best performances of the simple PS-classifier and the FCPS-classifier is quite low (about 2%) for Iris data classification with four-dimensional feature vectors. It is about 6% for wine data classification, which has 13 features for each pattern and the difference is significant: 10% for radar targets classification, whose feature space dimension is 128.

5. Conclusion and discussion

Powerfulness and effectiveness of PSO algorithm, specially in high dimensional spaces, are motivations to design a swarm intelligence based classifier, using the basic PSO to

obtain the decision hyperplanes in the feature space. Thus, the proposed classifier is a *basic* one. Now using the idea, anyone may construct the modified versions of the proposed classifier based on the many other kinds of PSO (maybe MPSO-classifier, QPSO-classifier, and so on).

Since inertia weight, cognitive parameter and social parameter are three important factors and have great effects on the search process of a PS-classifier, designing an intelligent fuzzy controller to adapt these parameters was considered. The fuzzy controlled PS-classifier (FCPS-classifier) was proposed for this purpose. The fuzzy controller rules were constructed due to a linguistic description of the roles of aforesaid parameters in the PSO search process. Experimental results for different kinds of data (with different feature space dimensions of 4, 13 and 128) indicate that for a given value of H (the number of hyperplanes), the designed IPS-classifier is able to efficiently approximate the decision hyperplanes. The performance of the classifier was also found to be comparable to or sometimes better than those of the k-NN and MLP classifiers.

Although in this paper a fuzzy structure has been selected as a candidate to develop an IPS-classifier, other kinds of intelligent controllers might be used to develop more efficient IPS-classifier instead of FCPS-classifier.

With regard to the system complexity, it may be noted that the complexity of an IPS-classifier is more than the complexity of a k-NN classifier. Regarding the timing requirements, it may be noted that the IPS-classifier takes a large amount of time *during training* like a MLP classifier; however, the time taken *during testing is very small*, because the decision hyperplanes have been already obtained in the training phase. Contrariwise, the k-NN classifier takes *significant amount of time for testing*. In fact the proposed IPS-classifier is an efficient offline classifier and may be applied for offline usages (offline signature recognition, face recognition, voice classification and etc.).

A theoretical analysis of the PS-classifier and designing an effective strategy to achieve a performance comparable to the Bayes classifier, which is an optimal conventional classifier (but with some limitations in usage due to need to important priori knowledge), are topic of future work.

Acknowledgement

This research has been supported by the Iranian Telecommunication Research Center (ITRC) through Grant 500-6618.

References

[1] J. Kennedy, R.C. Eberhart, Particle swarm optimization, Proc. IEEE Int. Conf. Neural Networks IV (1995) 1942–1948.
[2] Z. He, C. Wie, L. Yang, X. Gao, S. Yao, R.C. Eberhart, Y. Shi, Extracting rules from fuzzy neural network by particle swarm optimization, Proc. IEEE Int. Conf. Evol. Comput., 1998, pp. 74–77.
[3] H. Yoshida, K. Kawata, Y. Fukuyama, Y. Nakanishi, A particle swarm optimization for reactive power and voltage control considering voltage security assessment, IEEE Trans. Power Syst. 15 (2000) 1232–1239.
[4] D.W. Van der Merwe, A.P. Engelbrecht, Data clustering using particle swarm optimization, Proceedings of the 2003 Congress on Evolutionary Computation, 2003, pp. 215–220.
[5] Yan Yang, M. Kamel, Clustering ensemble using swarm intelligence, Proceedings of the 2003 IEEE Swarm Intelligence Symposium, SIS'03, 2003, pp. 65–71.
[6] Y. Shi, R.C. Eberhart, Fuzzy adaptive particle swarm optimization, Proceedings of the 2001 Congress on Evolutionary Computation, 2001, pp. 101–106.

[7] W. Zhang, Y. Liu, M. Clerc, An adaptive PSO algorithm for reactive power optimization, Proc. Int. Conf. Adv. Power Syst. Control, Oper. Manage. (2003) 302–307.
[8] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (1) (2002) 58–73.
[9] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, Proc. IEEE Int. Conf. Evol. Comput. (1998) 69–73.
[10] M. Clerc, The swarm and the queen: towards a deterministic and adaptive particle swarm optimization, Proceedings of the 1999 Congress on Evolutionary Computation, 1999, pp. 1951–1957.
[11] B. Al-kazemi, C.K. Mohan, Multi-phase generalization of the particle swarm optimization algorithm, Proceedings of the 2002 Congress on Evolutionary Computation, vol. 1, 2002, pp. 489–494.
[12] N. Higashi, H. Iba, Particle swarm optimization with Gaussian mutation, Proceedings of the 2003 IEEE Swarm Intelligence Symposium, SIS'03, 2003, pp. 72–79.
[13] S. Yang, M. Wang, L. Jiao, A quantum particle swarm optimization, Proceedings of the 2004 Congress on Evolutionary Computation, vol. 1, 2004, pp. 320–324.
[14] Y. Zheng, L. Ma, L. Zhang, J. Qian, On the convergence analysis and parameter selection in particle swarm optimization, Proceedings of the Second International Conference on Machine Learning and Cybernetics, 2003, pp. 1802–1807.
[15] B.R. Secrest, G.B. Lamont, Visualizing particle swarm optimization—Gaussian particle swarm optimization, Proceedings of the 2003 IEEE Swarm Intelligence Symposium, SIS'03, 2003, pp. 198–204.
[16] F. Van den Berg, An analysis of particle swarm optimizers, Ph.D. Thesis, Department of Computer Science, University of Pretoria, South Africa, 2002.
[17] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, Proceedings of the 1999 Congress on Evolutionary Computation, 1999, pp. 1945–1950.
[18] R.C. Eberhart, Y. Shi, Tracking and optimizing dynamic systems with particle swarms, Proceedings of the 2001 Congress on Evolutionary Computation, 2001.
[19] E. Ozcan, C. Mohan, Particle optimization: Surfing the waves, Proceedings of the International Congress on Evolutionary Computation, 1999, pp. 1939–1944.
[20] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, New York, 1972.
[21] Y. Shi, R.C. Eberhart, Experimental study of particle swarm optimization, The Fourth Multiconference on Systems, Cybernetics and Informatics, Orlando, Florida, USA, 2000.
[22] R.A. Fisher, The use of multiple measurements in taxonomic problems, Ann. Eugen 7 (1936) 179–188.
[23] University of California, Irvine, via anonymous ftp ftp.ics.uci.edu/pub/machine-learning-databases.
[24] M.R. Bell, R.A. Grubbs, JEM modeling and measurement for radar target identification, IEEE Trans. Aerosp. Electron. Syst. 29 (1) (1993) 73–87.
[25] Jane's Series, All the World Airplanes, 1986.
[26] J. Martin, B. Mulgrew, Analysis of the theoretical radar returned signal from aircraft propeller blades, Proc. IEEE Int. Radar Conf., 1990, pp. 569–572.