Neurocomputing ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Particle swarm optimization for prototype reduction

Loris Nanni *, Alessandra Lumini

DEIS, IEIIT—CNR, Università di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy

ARTICLE INFO

Article history:
Received 28 June 2007
Received in revised form
29 February 2008
Accepted 11 March 2008
Communicated by T. Heskes

Keywords:
Particle swarm optimization
Prototype reduction
Nearest neighbor

ABSTRACT

The problem addressed in this paper concerns the prototype reduction for a nearest-neighbor classifier. An efficient method based on particle swarm optimization is proposed here for finding a good set of prototypes. Starting from an initial random selection of a small number of training patterns, we generate a set of prototypes, using the particle swarm optimization, which minimizes the error rate on the training set. To improve the classification performance, during the training phase the prototype generation is repeated N times, then each of the resulting N sets of prototypes is used to classify each test pattern, and finally these N classification results are combined by the "vote rule".

The performance improvement with respect to the state-of-the-art approaches is validated through experiments with several benchmark datasets.

1. Introduction

Prototype reduction techniques are concerned with reducing the number of training vectors (prototypes) to be used, typically in conjunction with a nearest-neighbor (NN) classifier, for classification purpose. The aim of these approaches is to find out a minimal set of objects or prototypes to represent a training set such that the performance of a classifier built on the reduced design set is approximately as well as (or better than) that of a classifier built on the original dataset. Prototype reduction has been explored for various purposes, and has resulted in the development of many algorithms, which are usually divided into two categories: prototype *selection* (or *extraction*) and prototype *generation*. The first class concerns the identification of an optimal subset of representative objects from the original data, while the latter involves the creation of an entirely new set of objects. The interested reader can see [1] for a comprehensive overview of supervised reduction approaches.

A recent and very performing approach for prototype reduction, called *learning prototypes and distances* (LPD) [10], is based on the simultaneous search of a reduced set of prototypes and a suitable local metric for these prototypes. In fact, the use of an appropriately trained distance measure or metric that can be *global* (the same for the whole feature space), *class dependent* (a different metric for each class) or *local* (a different metric measure for each prototype), is another common technique for improving the performance of a NN classifier [10,11]. LDP starts with an initial random selection of a small number of prototypes and iteratively adjusts their position and their local metric according to a rule that minimizes a suitable estimation of the classification error probability. In [10] the authors show that LPD outperforms several state-of-the-art approaches based on learning vector quantization [8].

In [13] the authors have carried out an empirical study of the performance of four representative evolutionary algorithms models [12] for prototype selection. Moreover, they included a comparison between evolutionary algorithms and non-evolutionary pattern selection algorithms, reporting that the evolutionary methods outperform the non-evolutionary ones.

In this work, we investigate how a relatively novel evolutionary computation algorithm, particle swarm optimization (PSO) [5,6], can be applied to generate (not to select) an optimal set of prototypes. The social behavior of biological organisms, the movement of bird flocking, motivates the PSO algorithm. Each potential solution of an optimization problem is a bird (or "particle") with a given velocity flying trough the solution space. Each bird adjusts its flight according to its own flying experience and its companions' flying experience [16]. In [7] it is reported that often PSO outperforms genetic algorithms (GAs); moreover, the computation time of PSO is lower than that of GA.

The basic idea for applying PSO to prototype reduction is the following: a random small set of K training patterns is selected as initial solution. In this paper we propose to select $K = 5\%$ of the training set in order to obtain a substantial reduction. Then the position of these patterns, the "particles" of the optimization problem, is adjusted using PSO so that the classification error rate on the training set is minimized.

* Corresponding author.
E-mail address: lnanni@deis.unibo.it (L. Nanni).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.03.008

PSO is particularly attractive for prototype reduction in that particle swarms will discover the prototype positions as they fly within the solution space. The performance of the proposed algorithm is evaluated using several UCI datasets and compared with some state-of-the-art approaches for prototype reduction and classification based on the NN rule. Our experiments demonstrate that PSO has a strong search capability in the solution space and can discover optimal solutions quickly.

The paper is organized as follows: in Section 2 the new technique for prototype generation is reported, in Section 3 experimental results are presented. Finally, in Section 4 some concluding remarks are given.

2. PSO for prototype reduction

2.1. An introduction to PSO

PSO[1] is initialized with a population of P random solutions (P is the *swarm size*) as a GA, but instead of chromosomes they are called "particles". A particle i is represented by its position, a vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{is})$ in a S-dimensional space, and it can move in the solution space according to a velocity equation (see Eq. (1)). The best previous position of each particle according to the PSO fitness rule is recorded as $\mathbf{p}_i = (p_{i1}, p_{i2}, \ldots, p_{is})$; moreover, g records the index of the best particle position among all the $\mathbf{p}_i$, $i = 1, \ldots, P$. The velocity for particle i is a vector $\mathbf{v}_i = (v_{i1}, v_{i2}, \ldots, v_{is})$, of the same dimension of the vector that describe the particle's position; the position and the velocity of each particle change at each iteration of PSO according to the following update procedure:

$$v_i = wv_i + c_1 Rand()(p_i - x_i) + c_2 Rand()(p_g - x_i) \quad (1)$$

$$\mathbf{x}_i = \mathbf{x}_i + \mathbf{v}_i \quad (2)$$

where i varies from 1 to P ; c_1 and c_2 (named *acceleration constants*) represent the weighting of the stochastic acceleration terms that pull each particle toward the local ($\mathbf{p}_i$) and global ($\mathbf{p}_g$) best positions; *Rand*() is a random function in the range [0, 1]; w (named *inertia weight*) is a linear function of the number of iterations, which decreases along with the iterations, thus providing a balance between global and local exploration. The inertia weight varies from w_{start} to w_{end} according to the following equation:

$$w = \frac{(w_{\text{start}} - w_{\text{end}})(MAXITER - Iter)}{MAXITER + w_{\text{end}}} \quad (3)$$

where *MAXITER* is the maximum number of PSO iterations and *Iter* is the current iteration.

Moreover, the particles' velocities in each dimension are limited to a maximum velocity $v_{\max}$, which determines the maximum dimension of the steps through the solution space; $v_{\max}$ is a parameter of the optimization procedure that allows to setting of the maximum extent allowed for movements inside the solution space.

Eq. (1) is divided into three parts: the first part provides each particle with a "memory" of its last velocity, which decreases at each iteration (according to Eq. (3)), the second part is the "cognition" part, which represents the optimization of a given particle according to its own flying experience, the third part is the "social" part, which represents the optimization of a given particle according to the companions' flying experience.

[1] It is implemented as in PSO MATLAB TOOLBOX; it is available at ⟨http://psotoolbox.sourceforge.net⟩.

The algorithm for PSO optimization consists of an iterative updating of the swarm positions according to Eqs (1) and (2) on the basis of the best particle positions $\mathbf{p}_i$ calculated with respect to the fitness function. The search stops after *MAXITER* iterations or if no improvement is reached in the last iteration.

2.2. System overview

Given a training set containing T samples, where a generic training vector is $\mathbf{y} \in \mathbb{R}^M$, our aim is to find an optimal reduced set of prototypes of a prefixed cardinality K.

In order to obtain an optimal prototype reduction by PSO, we represent each particle as a set of K prototypes. A particle is a vector of length $S = KM$ given by the concatenation of K prototypes. The K prototypes used to initialize each particle are randomly extracted among the training patterns; in this work, we run PSO with $P = 20$ particles. The fitness function of PSO is simply the minimization of the classification error on the training set. The training phase ends after $MAXITER = 250$ iteration and the best position $\mathbf{p}_g$ is the concatenation of the K searched prototypes.

Since this prototype generation method is based on a random selection we repeat the generation step N times and we train N different classifiers to be combined by "vote rule". In this way the number of prototypes to be stored is KN, which is anyway lower than the cardinality T of the original training set. During the test phase the unknown pattern is compared with all the N set of prototypes according to the NN rule; then the final decision is obtained by selecting the most voted class (vote rule). The pseudo-code of the method, divided in training and test phases, is reported in Fig. 1 and a scheme of our approach is depicted in Fig. 2. In the psudo-code, in Fig. 1, we suppose that we receive in input for the training phase a training set **TR** containing T labeled samples $\mathbf{y}_i$ and for the test phase a set **TE** of E samples $\mathbf{x}_i$. During the training, the T samples are processed N times by means of the function PROTOPSO() in order to obtain a set **PR** of K labeled prototypes. During the test the unknown samples from the test set are assigned to a class according to a given set of prototypes by means of the function CLASSIFY(); then the final decision is obtained by applying the function VOTERULE() to the set of labels Λ predicted by the N classifiers.

3. Experiments

We perform experiments in order to (i) evaluate the choice of some critical parameters for our PSO-based approach, (ii) compare

```
TRAINING PHASE
for j=1:N
        PR(j) = PROTOPSO(TR);
end

TEST PHASE
for each x_i ∈ TE
        for j=1:N
                Λ(j)=CLASSIFY(x_i, PR(j));
        end
        λ_i=VOTERULE(Λ);
end
```

Fig. 1. Pseudo-code of the proposed method.

Please cite this article as: L. Nanni, A. Lumini, Particle swarm optimization for prototype reduction, Neurocomputing (2008), doi:10.1016/j.neucom.2008.03.008

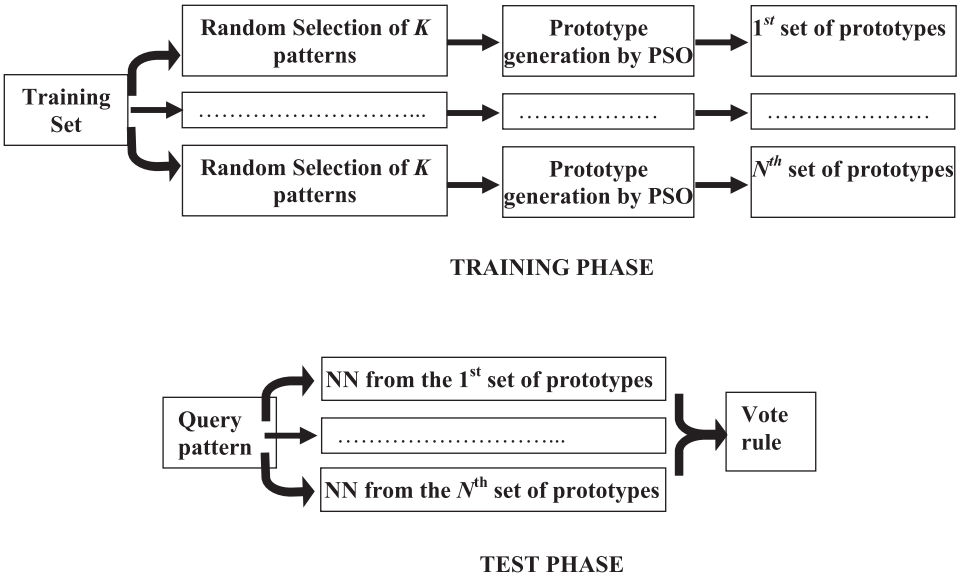

Fig. 2. Training and test phases of the proposed system.

Table 1
Characteristics of the datasets used in the experimentation: number of attributes (M), number of samples (T), and number of classes (C)

Dataset	M	T	C
IO	34	351	2
HEART	13	150	2
PIMA	8	768	2
BR	9	699	2
WINE	13	178	3
HIV	50	362	2
SPECTF	44	267	2

Table 2
Tested configurations of the parameters of PSO

Parameter	Configuration		
	First	Second	Third
v_{max}	0.25	0.05	0.1
C_1	1	1	3
C_2	3	3	1
w_{start}	1.5	0.5	2.5
w_{end}	0.5	0.15	1

the novel approach with another state-of-the-art method for prototype reduction (LPD), and (iii) compare the classification performance of our method with other NN-based classifiers. All the experiments have been conducted on six benchmark datasets from the UCI Repository[2] (Ionosphere (**IO**), Heart (**HE**), Pima Indians Diabetes (**PI**), Wisconsin Breast Cancer Databases (**BR**), Cardiac Single Proton Emission Computed Tomography (**SPECTF**), and Wine (**WI**)), and on the HIV protease dataset[3] (**HIV**) [14]. A summary of the characteristics of these datasets (number of attributes, number of samples, number of classes) is reported in Table 1. As suggested by many classification approaches, all the datasets have been normalized between 0 and 1. To minimize possible misleading caused by training data, results have been obtained using a twofold cross-validation on each dataset and averaged over 10 experiments.

3.1. Internal evaluation

Due to computational issues it was unfeasible to carry out a grid optimization of the PSO parameters; therefore we performed an heuristic setting, reporting the results for three different configuration of the PSO parameters (Table 2), which gained good performance.

In Table 3 the performance of our method (*PSO*(*N*)) using the configurations of Table 2 is reported; N denotes the number of classifiers combined by vote rule.

Since the third configuration allows us to obtain the best average results the following experiments have been performed fixing such parameter values.

As further experiment, we test the importance of the parameter *MAXITER* representing the number of iterations of PSO. In Fig. 3 we plot the error rate on the PIMA dataset (left) and on the IONO dataset (right) obtained by our method *PSO*(5), varying the number of iterations used in the PSO optimization algorithm. It is interesting to note that in both the datasets the performance reaches its maximum and becomes quite stable with a number of iterations between 100 and 250, over 250 iterations the performance decreases again, probably due to the overfitting on the training set.

In Table 4, the performance (in terms of error rate) for different values of the number N of combined classifiers is reported: since *PSO*(11) and *PSO*(15) only slightly outperform *PSO*(5), while the performance of *PSO*(5) is undoubtedly better than *PSO*(1) and due to computational issues N will be fixed to five in all the following comparisons.

The results reported in Fig. 3 and in Tables 3 and 4 prove that our prototype generation method permits us to find a good discriminative set of prototypes and it is quite easy to find a parameter configuration that allows us to obtain good performance in several different classification problems.

[2] ⟨http://www.ics.uci.edu/mlearn/MLRepository.html⟩.
[3] ⟨http://idelnx81.hh.se/bioinf/data.html⟩.

Please cite this article as: L. Nanni, A. Lumini, Particle swarm optimization for prototype reduction, Neurocomputing (2008), doi:10.1016/j.neucom.2008.03.008

Table 3
Classification error (standard deviation) obtained by our approach varying the parameter configuration

Dataset	Configuration First		Second		Third	
	PSO(1)	*PSO(5)*	*PSO(1)*	*PSO(5)*	*PSO(1)*	*PSO(5)*
IO	13.0 (2.5)	10.7 (2)	14.0 (2.3)	11.9 (2.1)	11.1 (1.5)	**10.6** (**2.0**)
HEART	18.0 (2.3)	**15.0** (**1.6**)	16.0 (2)	15.7 (1.7)	21.0 (2.7)	16.0 (1.7)
PIMA	25.8 (2.4)	25.3 (1)	27.5 (2.2)	25.6 (0.8)	25.7 (2.4)	**24.9** (**0.6**)
BR	4.3 (1.5)	4.0 (1.4)	4.0 (1.4)	3.5 (0.8)	4.0 (1.4)	3.4 (0.6)
WINE	9.0 (2.5)	**4.0** (**1**)	6.0 (2.2)	4.6 (1.2)	4.5 (2.2)	**4.0** (**0.8**)
HIV	20.0 (3.8)	16.1 (3.1)	17.0 (3.5)	14.4 (3.1)	16.0 (3.1)	**13.1** (**3.0**)
SPECTF	22.9 (2)	20.3 (1.3)	21.0 (1.5)	**18.8** (**1.2**)	23.8 (1.8)	20.6 (1.4)

Bold results are the best results for each dataset.

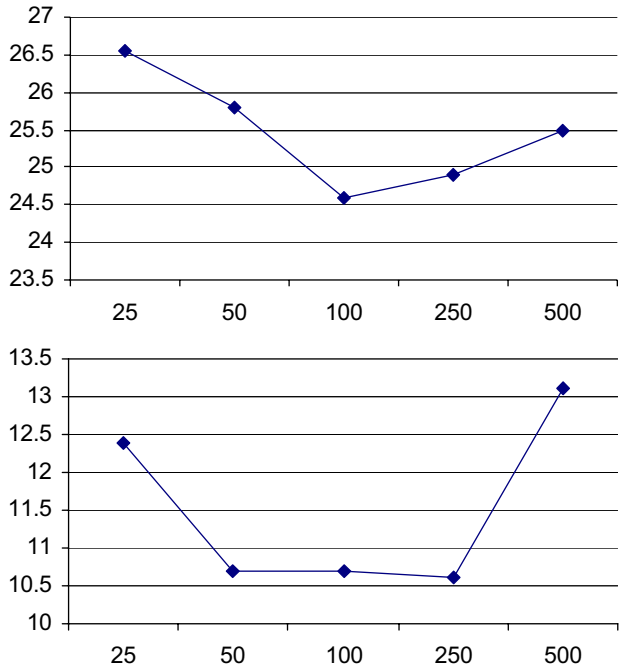

Fig. 3. Error rate obtained in the PIMA (left) and IONO (right) datasets by *PSO*(5) as a function of MAXITER, the number of iterations in the PSO optimization.

Table 4
Classification error (standard deviation) obtained by *PSO*(*N*), varying the value of *N*

Dataset	*PSO*(1)	*PSO(5)*	*PSO(11)*	*PSO(15)*
IO	11.1 (1.5)	10.6 (2.0)	**10.0 (1.5)**	10.0 (**1.5**)
HEART	21.0 (2.7)	16.0 (1.7)	15.4 (1.7)	**14.9 (1.6)**
PIMA	25.7 (2.4)	**24.9 (0.6)**	24.9 (**0.6**)	25.0 (0.6)
BR	4.0 (1.4)	3.4 (0.6)	**3.2 (0.5)**	3.3 (0.5)
WINE	4.5 (2.2)	**4.0 (0.8)**	4.0 (**0.9**)	4.0 (**0.8**)
HIV	16.0 (3.1)	13.1 (3.0)	**11.4 (2.8)**	11.4 (**2.9**)
SPECTF	23.8 (1.8)	20.6 (1.4)	**20.3 (1.4)**	20.4 (1.4)

Bold results are the best results for each dataset.

As a further experiment we investigated the relationship (i.e. the error independence) among the classifiers obtained using the ensemble *PSO*(5). It is well known in the literature [9] that combining "independent" classifiers permits dramatic reduction of error rate obtained by a "stand-alone" classifier. Error independence is an important property in combining classifiers; therefore an evaluation of the independence of the classifiers trained varying the prototype set is calculated using Yule's *Q*-statistic [9]. For two classifier D_i and D_j the *Q*-statistic a posteriori measure is defined as

$$Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}$$

where N^{ab} is the number of instances in the test set, classified correctly ($a = 1$) or incorrectly ($a = 0$) by the classifier D_i, and correctly ($b = 1$) or incorrectly ($b = 0$) by the classifier D_j. *Q* varies between −1 and 1; $Q_{i,j} = 0$ for statistically independent classifiers. Classifiers that tend to recognize the same patterns correctly will have $Q > 0$, and those that commit errors on different patterns will have $Q < 0$. In Table 5 the average *Q*-statistic among the classifiers in *PSO*(5) is reported: these tests show that where *PSO*(5) permits obtaining a pool of "enough independent" classifiers (e.g. in HIV, IO) there is a higher performance improvement with respect to the stand-alone approach (*PSO*(1), see Table 3).

3.2. Comparison with LPD

In this section a comparison between our PSO approach and LPD [10], one of the most recent state-of-the-art method for prototype reduction, is performed. Where not explicitly stated the number of the training patterns selected as starting prototypes for LPD is set to $K = 5\%$ of the dimensionality *T* of the training set (as in [10]). Please note that LPD is based on the simultaneous search of a reduced set of prototypes and a suitable local metric for these prototypes, while our method does not search a local metric since it is based on the simple Euclidean distance.

In order to make the comparison fair also for *PSO*(5), which is an ensemble, a modified version of LPD is also evaluated. The ensemble, named *LPD*(*N*), is obtained by combining the *N* vote rule LPD classifiers, each trained on a different prototype set (from different executions of LPD with new random initializations). In Table 6 the results of the comparison in terms of classification error are reported, showing that for all the tested problems *PSO*(5) outperforms the other classifiers. Moreover it is clear that the use of an ensemble allows both methods to improve the performance.

Finally, we test the importance of the cardinality *K* of the set of prototypes: In Table 7 we compare the performance obtained with

Table 5
Average *Q*-statistic among the classifiers in $PSO_3(5)$

Dataset	*Q*-statistic
IO	0.86
HEART	0.94
PIMA	0.88
BR	0.98
WINE	0.87
HIV	0.82
SPECTF	0.90

Table 6
Classification error (standard deviation): comparison between PSO and LPD

Dataset	*LPD*(1)	*LPD(5)*	*PSO(1)*	*PSO(5)*
IO	15.1 (3.7)	10.8 (2.0)	11.1 (1.5)	**10.6** (2.0)
HEART	16.8 (3.2)	17.0 (3.1)	21.0 (2.7)	**16.0 (1.7)**
PIMA	28.5 (1.5)	26.4 (1.0)	25.7 (2.4)	**24.9 (0.6)**
BR	3.7 (0.9)	3.5 (0.9)	4.0 (1.4)	**3.4 (0.6)**
WINE	4.1 (1.2)	4.5 (1.5)	4.5 (2.2)	**4.0 (0.8)**
HIV	16.7 (2.7)	18.3 (2.1)	16.0 (3.1)	**13.1** (3.0)
SPECTF	21.7 (2.5)	21.0 (3.5)	23.8 (1.8)	**20.6 (1.4)**

Bold results are the best results for each dataset.

Please cite this article as: L. Nanni, A. Lumini, Particle swarm optimization for prototype reduction, Neurocomputing (2008), doi:10.1016/j.neucom.2008.03.008

Table 7
Classification error (standard deviation) obtained varying the value of K

Dataset	*PSO*(5)		*W-PSO(5)*		*LPD(5)*	
	$K = 0.05T$	K = 0.025T	K = 0.05T	K = 0.025T	K = 0.05T	K = 0.025T
IO	10.6 (2.0)	10.9 (2.2)	**8.4** (1.9)	8.4 (1.9)	10.8 (2.0)	12.2 (2.2)
HEART	16.0 (1.7)	**14.9** (1.5)	17.0 (2.0)	16.0 (1.9)	17.0 (3.1)	16.0 (3.0)
PIMA	**24.9 (0.6)**	25.1 **(0.6)**	25.2 (0.8)	25.0 (0.8)	26.4 (1.0)	25.2 (1.0)
BR	**3.4 (0.6)**	3.6 (0.7)	3.7 (0.9)	3.6 (0.9)	3.5 (0.9)	3.6 (1.1)
WINE	**4.0 (0.8)**	**4.0 (0.8)**	4.4 (1.0)	4.5 (1.0)	4.5 (1.5)	**4.0** (1.4)
HIV	13.1 (3.0)	12.4 (2.9)	11.3 (3)	**11.0** (2.8)	18.3 **(2.1)**	20.6 (2.5)
SPECTF	20.6 (1.4)	**20.0 (1.3)**	20.5 (1.5)	21.0 (1.6)	21.0 (3.5)	21.5 (3.2)

Bold results are the best results for each dataset.

Table 8
Classification error (standard deviation) obtained by several approaches

Dataset	*PSO*(5)	*LPD*(5)	*NN*	*CNN*	*GA*
IO	**10.6** (2.0)	10.8 (2.0)	14.1 (1.0)	11.9 (1.1)	15.1 (2.0)
HEART	**16.0 (1.7)**	17.0 (3.1)	21.6 (2.9)	20.3 (2.6)	22.3 (2.5)
PIMA	**24.9 (0.6)**	26.4 (1.0)	30.5 (2.9)	31.2 (1.8)	26.2 (0.8)
BR	**3.4** (0.6)	3.5 (0.9)	4.4 **(0.6)**	5.2 (0.6)	4.6 (0.8)
WINE	**4.0** (0.8)	4.5 (1.5)	5.3 (1.3)	5.9 (0.5)	4.7 (1.3)
HIV	**13.1** (3.0)	18.3 (2.1)	24.5 (3.9)	13.1 (2.2)	17.0 (3.0)
SPECTF	**20.6 (1.4)**	21.0 (3.5)	30.4 (2.9)	26.5 (3.1)	25.0 (3.0)
WSR test	–	WIN	WIN	WIN	WIN
Training time, PIMA (s)	15	250	–	**2.6**	300
Test time, PIMA (s)	**0.58**−005	0.77−005	1.84−005	2.3−005	0.6−005

In the last three rows: results of the Wilcoxon signed-rank test between *PSO*(5) and the method reported in column, training and test times in seconds.
Bold results are the best results for each dataset.

$K = 0.05T$ (default value) and $K = 0.025T$. In these experiments a modified version of *PSO* is also tested, named *W-PSO*, which, similarly to LPD, learns a local metric, besides the set of prototypes. We use exactly the same local metric proposed in [10]: first the prototype set is learned, then the weights of the metric are obtained using PSO with objective function the minimization of the error rate on the training set.

It is interesting to note that using only 2.5% of the training patterns for generating the reduced set of prototypes, the error rate obtained by *PSO*(5) is anyway very low. As *W-PSO* is concerned, the low improvement in terms of accuracy does not justify the augmented complexity of the approach.

3.3. Comparison with other NN-based classifiers

This subsection reports a comparison ofour new method with other NN-based classifiers. Table 8 reports the error rate and the standard deviation (in parentheses) obtained by the following approaches:

- *PSO*(5), the proposed method;
- *LPD*(5), the learning prototype end distance method [10];
- *NN*, the simple 1-NN classifier [4];
- *CNN*, the center-based NN [3] recent NN-based classifier;
- *GA*, a GA [13] for prototype selection;

The proposed method always presents the lowest error rate of the tested approaches; moreover, *PSO*(5) is also stable as demonstrated by a quite low standard deviation, in particular if compared with the one of *LPD*(5) (the best of the remaining approaches).

The last but two line of Table 8 reports the result of the statistical Wilcoxon signed-rank test [2] calculated between *PSO*(5) and the method reported in column. The null hypothesis is that there is no difference between the accuracies of the two classifiers. We reject the null hypothesis (level of significance 0.05) and accept that the two classifiers have significant different accuracies. From the analysis of the experimental results is clear the advantage of the proposed approach with respect to the considered NN methods evaluated in this work it. The results of the Wilcoxon signed-rank test, which is considered [2] the best statistical measure to compare classifiers, demonstrate our thesis: *PSO*(5) wins against the other tested approaches.

The last two lines of Table 8 report the training and test times[4] of the considered approaches on the PIMA dataset; *PSO*(5) requires a training time that is obviously higher than a simple NN classifier; anyway it is much faster than other methods based on optimization (*LPD*(5), *GA*); moreover its test time is the lowest among the considered methods.

4. Conclusions

The problem addressed in this paper is the prototype reduction for a NN classifier. We have proposed a method based on PSO for the optimization of a good set of prototypes with respect to a given classification problem.

The NN classification has proved useful in obtaining a good behavior with unbounded number of prototypes; however, in many practical applications the presence of outliers and the requirement of reducing computational requirements of the classifier suggest the need of a prototype reduction in order to diminish the effect of outliers and the test time. Several works in the literature confirm that, with a substantial reduction in the prototype-set size, significant accuracy improvements over conventional, all-prototype NN approaches can be obtained.

[4] Non optimized MATLAB 7.4 code running on a 2.4 GHz pentium with 2GB RAM.

Please cite this article as: L. Nanni, A. Lumini, Particle swarm optimization for prototype reduction, Neurocomputing (2008), doi:10.1016/j.neucom.2008.03.008

We thought that PSO could be particularly suited for prototype reduction since particle swarms will discover the prototype positions as they fly within the solution space. In this work we have introduced an ad-hoc encoding of particles to solve the prototype reduction problem by PSO, and we have utilized advantage from the arbitrariness of the PSO initialization to design an ensemble of classifiers based on several prototype reduced sets obtained by different PSO executions.

The performance of the proposed algorithm has been evaluated on several UCI datasets and compared with some state-of-the-art approaches for prototype reduction and classification based on the NN rule. The reported experiments demonstrate that the approach named *PSO*(5) gains the lowest error rate against all the tested approaches and discover optimal solutions very quickly. The superiority of the proposed classifier is also revealed by the results of Wilcoxon signed-rank test.

As future work, we plan to improve the efficiency of our approach by implementing a feature weighting method Nanni and Lumini (i.e. a global metric) and/or a better local metric using an evolutionary method as in [15]. Another possible research direction concerns the study of an alternative encoding for PSO that allows optimization of the number of searched prototypes (which is fixed in the present method).

References

[1] J.C. Bezdek, L. Kuncheva, Nearest prototype classifier designs: an experimental study, Int. J. Intell. Syst. 16 (2001) 1445–1473.

[2] J. Demsar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.

[3] Q.-B. Gao, Z.-Z. Wang, Center-based nearest neighbour classifier, Pattern Recognition 40 (1) (2006) 346–349.

[4] P. Hart, The condensed NN rule, IEEE Trans. Inf. Theory 14 (3) (1968) 515–516.

[5] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, Perth, 1995, pp. 1942–1948.

[6] J. Kennedy, R.C. Eberhart, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, 1995, pp. 39–43.

[7] J. Kennedy, W.M. Spears, 1998. Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator, in: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 39–43.

[8] T. Kohonen, Self-Organizing Maps, third ed., Springer, New York, 2001.

[9] L.I. Kuncheva, C.J. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Mach. Learn. 51 (2003) 181–207.

[10] R. Parades, E. Vidal, Learning prototypes and distances: a prototype reduction technique based on nearest neighbor error minimization, Pattern Recognition 39 (2006) 180–188.

[11] C. Pedreira, Learning vector quantization with training data selection, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (1) (2006) 157–162.

[12] J. Ramón Cano, F. Herrera, M. Lozano, On the combination of evolutionary algorithms and stratified strategies for training set selection in data mining, Appl. Soft Comput. 6 (2006) 323–332.

[13] J. Ramón Cano, F. Herrera, M. Lozano, Using evolutionary algorithms as instance selection for data reduction in KDD: an experimental study, IEEE Trans. Evol. Comput. 7 (6) (2003) 561–575.

[14] T. Rögnvaldsson, L. You, Why neural networks should not be used for HIV-1 protease cleavage site prediction, BioInformatics 20 (11) (2004) 1702–1709.

[15] M.A. Tahir, A. Bouridane, F. Kurugollu, Simultaneous feature selection and feature weighting using Hybrid tabu search/K-nearest neighbor classifier, Pattern Recognition Lett. 28 (4) (2007) 438–446.

[16] X. Wang, J. Yang, X. Teng, W. Xia, R. Jensen, Feature selection based on rough sets and particle swarm optimization, Pattern Recognition Lett. 28 (4) (2007) 459–471.

Loris Nanni received his Master Degree cum laude in 2002 from the University of Bologna, and on April 26, 2006 he received his Ph.D. in Computer Engineering at DEIS, University of Bologna. His research interests include pattern recognition and biometric systems (fingerprint classification and recognition, signature verification, face recognition).

Alessandra Lumini received a degree in Computer Science from the University of Bologna, Italy, on March 26, 1996. In 1998 she started her Ph.D. studies at DEIS, University of Bologna and in 2001 she received her Ph.D. degree for her work on "Image Databases". Now she is an Associate Researcher at University of Bologna. She is a member of the BIAS Research Group at the Department of Computer Science of the University of Bologna (Cesena). She is interested in biometric systems (particularly fingerprint classification), multi-dimensional data structures, digital image watermarking and image generation.

Please cite this article as: L. Nanni, A. Lumini, Particle swarm optimization for prototype reduction, Neurocomputing (2008), doi:10.1016/j.neucom.2008.03.008