Research Article

A Constructive Data Classification Version of the Particle Swarm Optimization Algorithm

Alexandre Szabo and Leandro Nunes de Castro

Natural Computing Laboratory, Mackenzie University, Rua da Consolação 930, 01302-907 São Paulo, Brazil

Correspondence should be addressed to Leandro Nunes de Castro; lnunes@mackenzie.br

Received 2 October 2012; Revised 1 November 2012; Accepted 8 November 2012

Academic Editor: Baozhen Yao

Copyright © 2013 A. Szabo and L. N. de Castro. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The particle swarm optimization algorithm was originally introduced to solve continuous parameter optimization problems. It was soon modified to solve other types of optimization tasks and also to be applied to data analysis. In the latter case, however, there are few works in the literature that deal with the problem of dynamically building the architecture of the system. This paper introduces new particle swarm algorithms specifically designed to solve classification problems. The first proposal, named Particle Swarm Classifier (PSClass), is a derivation of a particle swarm clustering algorithm and its architecture, as in most classifiers, is pre-defined. The second proposal, named Constructive Particle Swarm Classifier (cPSClass), uses ideas from the immune system to automatically build the swarm. A sensitivity analysis of the growing procedure of cPSClass and an investigation into a proposed pruning procedure for this algorithm are performed. The proposals were applied to a wide range of databases from the literature and the results show that they are competitive in relation to other approaches, with the advantage of having a dynamically constructed architecture.

1. Introduction

Data classification is one of the most important tasks in data mining, which is applied to databases in order to label and describe characteristics of new input objects whose labels are unknown. Generally, this task requires a classifier model, obtained from samples of objects from the database, whose labels are known *a priori*. The process of building such a model is called *learning* or *training* and refers to the adjustment of parameters of the algorithm so as to maximize its generalization ability.

The prediction for unlabeled data is usually done using models generated in the training step or some lazy learning mechanism able to compare objects classified *a priori* with new objects yet to be classified. Thus, not all algorithms generate an explicit classification model (classifier), as is the case of k -NN (k -nearest neighbors) [1, 2], and Naïve Bayes [3], which use historical data in a comparative process of distance or probability estimation, respectively, to classify new objects. Examples of algorithms that generate an explicit classification model are the decision trees [4, 5], artificial neural networks

[6, 7], and learning vector quantization (LVQ) algorithms [8–11].

This paper proposes two algorithms for data classification—Particle Swarm Classifier (PSClass) and its successor Constructive Particle Swarm Classifier (cPSClass), both based on the particle swarm optimization algorithm (PSO) [12]. These algorithms were developed from adaptations of the PSO and other bioinspired techniques, and were evaluated in seven databases from the literature. Their performance was compared with that of other swarm algorithms and also with some well-known methods from the literature, such as k -NN, naïve Bayes, and an MLP neural network.

In the PSClass algorithm, two steps are necessary to construct the classifier. In the first step, a number of prototypes are positioned, in an unsupervised way, on regions of the input space with some density of data; for this, the Particle Swarm Clustering (PSC) [13] algorithm is used. In the next step, the prototypes are adjusted by an LVQ1 method [14] in order to minimize the percentage of misclassification. Thus, the PSCClass automatically positions the prototypes in the respective classes of objects, defining the decision boundaries

between classes and increasing the efficiency of the algorithm during the construction of the classifier.

The cPSClazz, in turn, improved its unsupervised step by inserting mechanisms inspired by the immune system to automatically build the classifier model, more specifically, to automatically set the number of cells (prototypes) in the swarm. At this step, the PSC has been replaced by the Constructive Particle Swarm Clustering (cPSC) [15], which uses the clonal selection principle [16] to iteratively control the number of particles in the swarm. Furthermore, a pruning phase was introduced so as to avoid the explosion in the number of particles. Thus, cPSClazz eliminates the need for the user to define the swarm size a priori, a critical parameter required for many data classification algorithms.

The paper is organized as follows. As the cPSClazz algorithm borrows ideas from swarm intelligence and the immune system, Section 2 is dedicated to a brief review of the biological concepts necessary for a proper understanding of the algorithm. Section 3 presents the PSC and cPSC algorithms, which form the basis for designing PSClazz and cPSClazz. In Section 4, two classifiers are proposed based on the PSO—PSClazz and cPSClazz. Section 5 provides a literature review emphasizing algorithms based on the PSO to solve classification problems and PSO versions with dynamic population. The PSClazz and cPSClazz algorithms are evaluated in Section 6 and a parametric sensitivity analysis for cPSClazz was performed to evaluate the growth of the swarm. The paper is concluded in Section 7, with discussions and proposals for future works.

2. Biological Background

Biological and behavioral mechanisms are some of the natural inspirations that motivate scientists and engineers to construct intelligent computational tools. One of the pioneer computational bioinspired tools was proposed by McCulloch and Pitts [17], with their logic model of the neuron. After that, several other lines of research have emerged, with new bioinspirations, such as evolutionary computing with the Darwinian laws of evolution [18–20], swarm intelligence [12, 21–23] inspired by the emergent behavior of social agents (most often insects), and artificial immune systems, inspired by the vertebrate immune system [24, 25]. These four approaches constitute one of the three major areas of natural computing [26]. The following sections briefly review swarm intelligence concepts and their main lines of research, as well as an overview of the vertebrate immune system and its main defense mechanisms.

2.1. Swarm Intelligence. Collective systems (swarms) are composed of agents that interact with each other and with the environment in order to achieve a common objective. These agents can be formed by a flock of birds, a school of fish, or a colony of insects, which are able to learn from their own experience and with the social influence and adapt to changes in the environment where they live [27]. These agents individually have limited cognitive abilities that allow them to perform only simple tasks. However, when put together and

interacting with one another, they can perform substantially complex tasks [28]. The emergent behavior of this social interaction is called swarm intelligence.

This terminology was first used in the work of Beni and Wang [21], which described the behavior of a group of robots that interacted with each other in a particular environment, respecting a set of local rules inspired by the behavior of ants. According to Kennedy et al. [29], any collective behavior, like a flock of birds or an immune system, can be named a swarm.

There are basically two approaches in Swarm Intelligence: the works based on the behavior of social insects [30] and those based on the human ability to process knowledge [31]. In both lines of research, there is a group of agents that interact with one another and with the environment.

Among the swarm intelligence algorithms, a great deal of attention has been given to the PSO algorithm, introduced by Kennedy and Eberhart [12]. PSO was inspired by the social behavior of flocks of birds or schools of fish to solve complex optimization problems. It uses a population of solutions, called particles, which interact with one another exchanging experience in order to optimize its ability within a certain environment. The main bioinspiration in the PSO is that one behaves according to its own past experience and that of other interacting agents. Since its introduction, PSO has also been improved and adapted to be applied to various tasks [32].

2.2. Immune System. Living organisms have cells and molecules that protect their body against the onslaught of disease-causing agents (*pathogens*). Specific cells and their mechanisms, such as identification, signaling, reproduction, and attack, are parts of a complex system called the immune system [24], responsible for keeping diseases at bay. The vertebrate immune system is divided mainly into *innate immune system* and *adaptive immune system*.

The innate immune system does not evolve, remaining the same throughout the life time of the individual. It recognizes many infectious diseases and is responsible for combating infectious agents while the adaptive immune system is preparing to act. The innate immune system, by itself, cannot remove most pathogens [24]. The main role of the innate immune system is to signal other immune system cells, since most pathogens do not directly stimulate the cells of the adaptive immune system.

The adaptive immune system is also called specific immune system, because some pathogens are recognized only by cells and molecules of the adaptive immune system. One of its main features is the ability to learn from infections and develop an immune memory, in other words, to recognize an antigen when it is presented recursively to the body. Thus, the adaptability of the adaptive immune system renders it more capable of recognizing and combating specific antigens each time it tries to reinfect the body.

The main components of the innate immune system are the macrophages and granulocytes, while in the adaptive immune system these are the lymphocytes. Both systems depend on the activity of white blood cells (leukocytes).

The organs that make up the immune system are called lymphoid organs consisting of the following two subsystems:

- (i) primary lymphoid organs are responsible for the production, growth, development, and maturation of lymphocytes.
- (ii) secondary lymphoid organs are sites in which lymphocytes recognize and fight pathogens.

Some lymphocytes are generated, developed, and matured in the bone marrow. These lymphocytes are called B cells. However, some lymphocytes generated in the bone marrow migrate to an organ called *thymus*, where they are matured and come to be called the T cells. The B cells and T cells are the lymphocytes of the immune system.

The main mechanisms of recognition and activation of the immune system are described by the *clonal selection principle* or the *clonal selection theory* [16], which explains how the adaptive immune system responds to pathogens. Only cells that recognize antigens are selected for reproduction, whilst the remainder die after some time due to a lack of stimulation.

The immune cells subjected to high concentrations of antigens (pathogens) are selected as candidates for reproduction. If the affinity between this cell receptor (called *antibody*) and an antigen of greater affinity to it exceeds a certain threshold, the cell is cloned. The immune recognition, thus, triggers the proliferation of antibodies as one of the main mechanisms of immune response to an antigenic attack, a process called clonal expansion. During clonal expansion B cells are subjected to an affinity proportional mutation process, resulting in variations in the repertoire of immune cells and, thus, antibodies, which are B-cell receptors free in solution.

The clonal selection theory is considered the core of the immune response system, since it describes the dynamics of the adaptive immune system when stimulated by disease-causing agents. Therefore, it is used in the design of adaptive problem solving systems [24] and was used in the design of the cPSC algorithm to be proposed in this paper.

3. Clustering Using Particle Swarm

The idea of using the PSO algorithm to solve clustering problems was initially proposed in [33], so that each particle corresponds to a vector containing the centroid of each group of the database. Since then, several clustering algorithms based on the PSO have been proposed [13, 15, 31, 33–56]. Among them, the PSC and cPSC algorithms, form the basis for the PSCl and cPSCl classification algorithms, respectively, proposed in this paper. In this section, the precursors of PSCl and cPSCl are reviewed.

3.1. Particle Swarm Clustering (PSC). The PSC, proposed in [13], is an adaptation of the PSO to solve data clustering problems. In the PSC, particles interact with one another and with the environment (database) so that they become representatives of a natural group from the database. The convergence criterion of the algorithm is determined by the stabilization of the path of the particles, and the number of particles in the swarm is initialized empirically. The dimension of a particle is given by the dimension of the input objects, where each vector position is an attribute of the object.

The main structural differences between the PSC and PSO algorithms are as follows.

- (i) In PSC, the particles altogether compose a solution to the data clustering problem.
- (ii) The PSC does not use an explicit cost function to evaluate the quality of the particles. Instead, the Euclidean distance is used as a measure to assess the dissimilarity between a particle and an object, and particles move around the space in order to represent statistical regularities of the input data.
- (iii) A self-organizing term, which moves the particle towards the input object, was added to the velocity equation.
- (iv) In the PSO algorithm, all the particles in the swarm are updated at each iteration. In the PSC, the particles to be updated are defined by input objects (i.e., only the *winner*—the one closest to the input datum—is updated according to (1) and (2)).

For each input object, there is a particle of greater similarity to it, obtained by the Euclidean distance between the particle and the object. This particle is called *winner*, and is updated following the proposed velocity equation (1) as

$$v_i(t+1) = \omega * v_i(t) + \varphi_1 \otimes (p_i^j(t) - x_i(t)) + \varphi_2 \otimes (g^j(t) - x_i(t)) + \varphi_3 \otimes (y^j - x_i(t)). \quad (1)$$

In (1), the parameter ω , called *inertia moment*, is responsible for controlling the convergence of the algorithm. The cognitive term, $\varphi_1 \otimes (p_i^j(t) - x_i(t))$, associated with the experience of the particle, represents the best particle's position $p_i^j(t)$, in relation to the input object so far, that is, the smallest distance between the input object (y^j) and the *winner* particle (x_i). The social term, $\varphi_2 \otimes (g^j(t) - x_i(t))$, is associated with the particle $g^j(t)$ closest to the input object, that is, the particle that had the smallest distance in relation to the input object (y^j) so far. The self-organizing term, $\varphi_3 \otimes (y^j - x_i(t))$, moves the particle towards the input object.

$$x_i(t+1) = x_i(t) + v_i(t+1). \quad (2)$$

Thus, the particles converge to the centroid of the groups, or regions of higher density of objects, becoming prototypes representatives of groups from the database.

The pseudocode of the PSC algorithm is described in Pseudocode 1.

Step 13 of Pseudocode 1 assigns a label (**PLABELS**) to each input object, which is given by a label (**CLABELS**) that represents the dominant class of objects for which each particle was the *winner*. Generally, in real world problems the correct labels are not known *a priori*. So, each label (**CLABELS**) must be given by each particle in the swarm.

Step 26 of Pseudocode 1 updates all those particles that did not move at iteration t . Thus, after all objects were presented to the swarm, the algorithm verifies whether some particle x_i did not *win* in that iteration ($x_i! = \text{win}$). If yes, then

```

1. procedure [X, V, P, g,  $\omega$ , PLABELS] = PSC (Y,  $v_{\max}$ ,  $m$ ,  $\omega$ , CLABELS)
2. Y//dataset
3. initialize X//initialize at random each particle  $x_i \in [0,1]$ 
4. initialize  $v_i$ //initialize at random each  $v_i \in [-v_{\max}, v_{\max}]$ 
5. initialize dist
6.  $t = 1$ 
7. while stopping criterion is not met
8.   for  $j = 1$  to  $n$ //for each input datum
9.     for  $i = 1$  to  $m$ //for each particle
10.      dist ( $i$ ) = distance ( $x_i, y^j$ )
11.    end for
12.     $I = \text{index}(\min(\mathbf{dist}))$ 
13.    [PLABELS] = label ( $x_I$ , CLABELS ( $y^j$ ))//predicted label
14.    if distance ( $x_I, y^j$ ) < distance ( $p_I^j, y^j$ )
15.       $p_I^j = x_I$ 
16.    end if
17.    if distance ( $x_I, y^j$ ) < distance ( $g^j, y^j$ )
18.       $g^j = x_I$ 
19.    end if
20.     $v_I(t+1) = \omega * v_I(t) + \varphi_1 \otimes (p_I^j(t) - x_I(t)) + \varphi_2 \otimes (g^j(t) - x_I(t)) + \varphi_3 \otimes (y^j - x_I(t))$ 
21.     $v_I \in [-v_{\max}, v_{\max}]$ 
22.     $x_I(t+1) = x_I(t) + v_I(t+1)$ 
23.     $x_I \in [0, 1]$ 
24.  end for
25.  for  $i = 1$  to  $m$ 
26.    if ( $x_i \neq \text{win}$ )//particles did not win
27.       $v_i(t+1) = \omega * v_i(t) + \varphi_4 \otimes (x_{\text{most\_win}} - x_i(t))$ 
28.       $v_i \in [-v_{\max}, v_{\max}]$ 
29.       $x_i(t+1) = x_i(t) + v_i(t+1)$ 
30.    end if
31.  end for
32.   $t = t + 1$ 
33.   $\omega = 0.95 * \omega$ 
34.  Test the stopping criterion
35. end while
36. end procedure

```

PSEUDOCODE 1: Particle Swarm Clustering (PSC).

these particles are updated in relation to the particle that was elected the winner more often at iteration t . Such particle is called $x_{\text{most_win}}$ (step 27 in Pseudocode 1), where φ_4 is a random vector within the interval (0,1).

3.2. Constructive Particle Swarm Clustering (cPSC). To dynamically determine the number of particles in PSC, Prior and de Castro [15] proposed a successor, called cPSC, which eliminated the need for the user to input the number of particles (prototypes) in the swarm. cPSC automatically finds a suitable number of prototypes in databases by employing strategies borrowed from the PSC algorithm with the addition of three new features inspired by the antibody network named ABNET [57]: growth of the particle swarm, pruning of particles, and automatic stopping criterion. Furthermore, the cPSC algorithm uses an affinity threshold (ϵ) as a criterion to control the growth of the swarm. The growth, pruning and stopping functions are described below and are evaluated at every two iterations.

3.2.1. Swarm Growth. The growth stage is based on the immune cell reproduction mechanism during clonal selection and expansion [16, 24], as described previously. The cells that are subjected to high concentrations of antigens are chosen as candidates to reproduce. If the affinity between the antibodies of these cells in relation to the antigens of higher affinities to them is greater than a threshold ϵ , then these cells are cloned.

These principles of selection and reproduction of antibodies inspired the design of the constructive particle swarm clustering algorithm. In the cPSC, particles are analogs to immune cells and objects from the database to antigens. The algorithm starts with only one particle (immune cell), with position and velocity initially random. At every two iterations the algorithm evaluates the necessity of growing the swarm as follows: the particle that was elected the winner more times (cell submitted to the highest concentration of antigen) is selected. The algorithm evaluates the degree of affinity between the particle and the object (antigen) of higher affinity to it, using threshold ϵ . If the affinity between them is greater

```

1. Procedure [X, V, P, g,  $\omega$ , PLABELS] = cPSC (Y,  $m$ ,  $\omega$ ,  $v_{\max}$ ,  $\varepsilon$ , CLABELS)
2. Y//dataset
3. initialize X//initialize at random only one particle  $x_i \in [0, 1]$ 
4. initialize V//initialize at random,  $v_i \in [-v_{\max}, v_{\max}]$ 
5. initialize dist
6.  $t = 1$ 
7. while stopping criterion is not met
8.   for  $j = 1$  to  $n$ //for each input datum
9.     for  $i = 1$  to  $m$ //for each particle
10.      dist ( $i$ ) = distance ( $x_i, y^j$ )
11.    end for
12.     $I = \text{index}(\min(\mathbf{dist}))$ 
13.    [PLABELS] = label ( $x_I$ , CLABELS ( $y^j$ ))//predicted label
14.    if distance ( $x_I, y^j$ ) < distance ( $p_I^j, y^j$ )
15.       $p_I^j = x_I$ 
16.    end if
17.    if distance ( $x_I, y^j$ ) < distance ( $g^j, y^j$ )
18.       $g^j = x_I$ 
19.    end if
20.     $v_I(t+1) = \omega * v_I(t) + \varphi_1 \otimes (p_I^j(t) - x_I(t)) + \varphi_2 \otimes (g^j(t) - x_I(t)) + \varphi_3 \otimes (y^j - x_I(t))$ 
21.     $v_I \in [-v_{\max}, v_{\max}]$ 
22.     $x_I(t+1) = x_I(t) + v_I(t+1)$ 
23.     $x_I \in [0, 1]$ 
24.  end for
25.  if mod ( $t, 2$ ) == 0
26.    Eliminate particles from the swarm if necessary
27.    Test the stopping criterion
28.    Clone particles if necessary
29.  end if
30.   $t = t + 1$ 
31.   $\omega = 0.95 * \omega$ 
32. end while
33. end procedure

```

PSEUDOCODE 2: Constructive particle swarm clustering.

than ε , then a new particle is created in the swarm. This new particle is positioned in the middle between the winner and the object of higher affinity to it.

3.2.2. Pruning of Particles. At every two iterations, the algorithm evaluates the need for pruning particles. If a particle has not moved at all in two iterations, then it is deleted from the swarm. A new step, called *suppression*, was added right after the pruning of particles step. If the particles are close to one another (Euclidean distance < 0.3), they are eliminated. It is a metaphor based on the immune system: cells and molecules recognize each other even in the absence of antigens. If the cell recognizes an antigen (*positive response*), then a clonal immune response is started; otherwise, (*negative response*) a *suppression*, which refers to the death of cells recognized as self, happens.

3.2.3. Stopping Criterion. At every two iterations the algorithm evaluates the stopping criterion by the average Euclidean distance between the current position and the position of the memory particles. Thus, the algorithm stops when this distance is less than or equal to 10^{-3} or 200 iterations.

The pseudocode of the cPSC algorithm is described as in Pseudocode 2.

4. Classification Using Particle Swarm

This paper proposes two data classification algorithms based on particle swarms: (1) PSClass, that uses the LVQ1 heuristics to adjust the position of prototypes generated by a clustering swarm-based algorithm; and (2) cPSClass, an improved version of PSClass that uses ideas from the immune system to dynamically determine the number of particles in the swarm.

The training process of PSClass consists of the iterative adjustment of the position of particles (prototypes). After training, there is a predictor model formed by a set of prototypes able to describe and predict the class to which a new input object from the database must belong. The testing step assesses the generalization capability of the classifier. At this stage, a number of test objects are presented to the classifier and their classes are predicted.

Two methods are combined to generate the predictor: the PSC algorithm, and an LVQ1 model. The LVQ1 heuristics was adopted for its simplicity and allows, through simple


```

1. procedure[X, PLABELS] = PSClass (Y,  $v_{\max}$ ,  $m$ ,  $\omega$ , CLABELS)
2. Y//dataset
3. CLABELS//correct_labels
4. [X, V, P, g,  $\omega$ , PLABELS] = PSC (Y,  $v_{\max}$ ,  $m$ ,  $\omega$ , CLABELS)
5. while stopping criterion is not met
6.   for  $j = 1$  to  $n$ //for each object
7.     for  $i = 1$  to  $m$ //for each particle
8.       dist ( $i$ ) = distance ( $x_i$ ,  $y^j$ )
9.     end for
10.     $I = \text{index}(\min(\mathbf{dist}))$ 
11.    if distance ( $x_I$ ,  $y^j$ ) < distance ( $p_I^j$ ,  $y^j$ )
12.       $p_I^j = x_I$ 
13.    end if
14.    if distance ( $x_I$ ,  $y^j$ ) < distance ( $g^j$ ,  $y^j$ )
15.       $g^j = x_I$ 
16.    end if
17.     $v_I(t+1) = \omega * v_I(t) + \varphi_1 \otimes (p_I^j(t) - x_I(t)) + \varphi_2 \otimes (g^j(t) - x_I(t)) + \varphi_3 \otimes (y^j - x_I(t))$ 
18.     $v_I \in [-v_{\max}, v_{\max}]$ 
19.    if (PLABELS ( $I$ ) == CLABELS ( $I$ ))
20.       $x_I(t+1) = x_I(t) + v_I(t+1)$ 
21.    else
22.       $x_I(t+1) = x_I(t) - v_I(t+1)$ 
23.    end if
24.  end for
25.   $\omega = 0.95 * \omega$ 
26.  Test the stopping criterion
27. end while
28. end procedure

```

PSEUDOCODE 3: Particle Swarm Classifier (PSClass).

procedures of position adjustment, the correction of misplaced prototypes in the data space [10].

Within PSClass, the PSC algorithm is run to find groups of objects in the database by placing the particles (prototypes) on the natural groups of the database. The number of particles must be informed by the user and this number should be at least equal to the number of the existing classes in the database. The algorithm places the prototypes in the input objects space in order to map each object in a representative prototype of each class. Thus, the algorithm generates a decision boundary between classes based on the prototypes that represent the classes. Then, the LVQ1 heuristics is used to adjust the position of the prototypes so as to minimize the classification error.

In its classification version, the PSC algorithm was modified such that the number of iterations for convergence is not determined by the user. Instead, its stopping criterion is defined by the stabilization of the prototypes around the input objects.

Two steps are required to generate the PSClass classifier.

- (i) *Unsupervised Step*. In this stage, the PSC algorithm is run in order to position the particles in regions of the input space that are representative of the natural clusters of data.
- (ii) *Supervised Step*. Some steps are added to the PSC algorithm such that the prototypes are adjusted by

the LVQ1 method so as to minimize the classification error, as shown in (3) and (4).

For each object j in the database, there is a prototype i with greater similarity to it, determined by a nearest neighbor method. This prototype is updated considering the PSC equations combined with the LVQ1 method as

$$x_i(t+1) = x_i(t) + v_i(t+1), \quad (3)$$

if the prototype x_i and the object y^j belong to the same class; and

$$x_i(t+1) = x_i(t) - v_i(t+1), \quad (4)$$

if the prototype x_i and the object y^j do not belong to the same class.

Thus, when a particle labels correctly an object from the database, the particle is moved toward this object (3); otherwise, it is moved in the opposite direction to the object (4).

The following pseudocode presents the supervised step of the PSClass classifier (see Pseudocode 3).

As in most data classification algorithms, the user must define the architecture of the system, for example, number of particles in the swarm or neurons in the neural network; some changes in the PSClass classifier were proposed so that it could dynamically determine the number of prototypes in the swarm, and, thus, the automatic construction of a classifier model, giving rise to cPSClass. The cPSClass algorithm was

inspired by the work of Prior and de Castro [15], with the proposal of the cPSC, discussed in Section 3.2.

Like its predecessor, PSClass, two steps are necessary to generate the cPSCClass classifier.

- (i) *Unsupervised Step*. In this stage, the cPSC algorithm is run in order to position the particles in regions of the input space that are representative of the natural clusters of data and also to determine a suitable number of particles in the swarm (Pseudocode 2).
- (ii) *Supervised Step*. Some steps are added to the PSC algorithm such that the prototypes are adjusted by the LVQ1 method so as to minimize the classification error, as shown in (3) and (4).

5. Related Works

As the contributions of this paper emphasize a constructive particle swarm classification algorithm, the related works to be reviewed here include the use of the PSO algorithm for data classification and PSO techniques with dynamic population.

5.1. PSO for Data Classification. There are several works in the literature involving data classification with PSO, such as [34–41]. These will be briefly reviewed in this section.

In [34], two approaches to the binary PSO are applied to classification problems: one called Pittsburgh PSO (PPSO) and the other called Michigan PSO (MPSO). In the Pittsburgh approach, each particle represents one or more prediction rules. Thus, a single particle is used to solve a classification problem. The classification is done based on the nearest neighbors rule (NN). In the Michigan PSO, by contrast, each particle represents a single prototype. Thus, all particles are used to solve a classification problem. A refinement of the MPSO is presented in [35] with the adaptive Michigan PSO (AMPPO), where the population is dynamic and the whole swarm is used to solve the problem. The fitness of each particle is used to calculate its growth probability.

In [36], the authors proposed an extension of the binary PSO, called DPSO, to discover classification rules. Each attribute may or may not be included in the resulting rule. An improvement of DPSO was proposed in [37], culminating in the hybrid algorithm PSO/ACO. The proposal starts with an empty set of rules, and for each class from the database it returns the best rule for the class evaluated.

Hybrid algorithms are common, as is the case of PSORS [38], which is used to cluster and classify objects. It combines the PSO, rough sets (RS), and a modified form of the Huang Index function to optimize the number of groups found in the database. The number of groups for each attribute of the particle is limited by a range defined by the Huang index, which is applied to the database. Attributes are fuzzified by the fuzzy *c*-means [42], and the index function is applied to each object to determine the group to which it belongs.

In [39] it was proposed a hybrid algorithm, named hybrid particle swarm optimization tabu search (HPSOTS), for selecting genes for tumor classification. The HPSOTS combines PSO with tabu search (TS) to maintain population

diversity and prevent deceptive local optimum. The algorithm initializes a population of individuals represented by binary strings. Ninety percent of the neighbors of an individual are assessed according to mechanisms from the modified discrete PSO [43, 44]. The algorithm selects a new individual of the neighborhood according to the tabu conditions and updates the population.

According to Wang et al. [40], many classification techniques, such as decision trees [4, 5] and artificial neural networks [6, 7], do not produce acceptable predictive accuracy when applied to real problems. In this sense, the authors proposed the use of the PSO algorithm [12] to discover classification rules. Their method initializes a population of individuals (rules) with the dimension given by the number of attributes of the objects evaluated. A fitness function is defined to evaluate the solution (classification rules) for the problem in question. The smaller the rule set, the easier to understand it.

The works presented in [40, 45] also cite the quality of the results produced when using classical techniques to solve real world problems. For Wang et al. [40], decision trees, artificial neural networks, and naive Bayes are some of the classic techniques applied to classification problems and they work well in linearly separable problems. The authors proposed a classification method based on a multiple linear regression model (MRLM) and the PSO algorithm, called the PSO-MRLM, which is able to learn the relationship between objects from a database and also express it mathematically. The MRLM technique builds a mathematical model able to represent the relationship between variables, associating the value of each independent variable with the value of the dependent variable. The set of equations (rules) contemplate this relationship using coefficients that act on each of the attributes of each rule. The proposed method uses the PSO algorithm to estimate the value of the coefficients.

A classifier based on PSO is proposed in [41] and applied to power systems operations. Pattern recognition based on PSO (PSO-PR) evaluates a condition of operation and predicts whether this is safe or unsafe. The first step to obtain the classifier is to generate the patterns (data) necessary to the training process. As the number of variables describing the power system state is very large, the next step in this process involves a feature selection procedure, responsible for eliminating redundant and irrelevant variables. In the next step PSO is used to minimize the percentage of misclassification.

Compared with the proposed PSClass and cPSCClass, none of the works available in the literature work by using a clustering algorithm followed by a vector quantization approach. The proposals here initially operate in a completely self-organized manner, and only after the particles are positioned in regions of the space that represent the input data, their positions are corrected so as to minimize the classification error. Despite these differences, in the present paper the performances of PPSO, MPSO, and AMPPO algorithms are compared with that of PSClass and cPSCClass.

5.2. PSO with Dynamic Population. The original PSO has also been modified to dynamically determine the number of particles in the swarm. According to [46], there are few

publications dealing with the issue of dynamic population size in PSO, and the main ones are briefly reviewed below.

In [47] two dynamic population approaches are proposed to improve the PSO speed: expanding population PSO (EP-PSO) and diminishing population PSO (DP-PSO). According to the experiments shown, both approaches reduce the run time of PSO by 60% on average.

In [48], it was proposed the dynamic population PSO (DPPSO), where the number of particles is controlled by a function that contains an attenuation item (responsible for reducing the number of particles) and a waving item (particles are generated to avoid local optimum and the ones considered inefficient die and are removed from the swarm) to control the population size.

In [46] it was proposed the dynamic multiobjective particle swarm optimization (DMOPSO) algorithm, which uses the particle growth strategy inspired by the algorithm incrementing multiobjective evolutionary (IMOEA) [49] in which particles of best fitness are selected to generate new particles.

In [50], the authors proposed the ladder particle swarm optimization (LPSO) algorithm, where the population size is determined based on its diversity.

In [51], two approaches to multiobjective optimization with dynamic population are proposed: dynamic multiobjective particle swarm optimization (DPSMO) and dynamic particle swarm evolutionary algorithms (DPSEA), which combine PSO and genetic algorithm mechanisms to regulate the number of particles in the swarm.

All PSO versions with dynamic population present growth strategies for the number of particles to ensure the diversity of solutions and pruning strategies to reduce the processing cost. Their applications, however, are focused on optimization problems, not on data classification problems, as proposed in the present paper. Therefore, no direct performance comparisons will be made with these methods.

6. Performance Assessment

The PSClazz and cPSClazz algorithms were implemented in MATLAB 7.0, and their performances were compared with those three algorithms based on the PSO: PPSO, MPSO and AMPSO, and also with the three well-known classification algorithms from the literature: naïve Bayes, k -NN and a multi-layer perceptron (MLP) neural network trained with the backpropagation algorithm [11]. The parametric configurations for the MLP were as follows: *learning rate* equals to 0.3, maximum number of *epochs* equals to 500, and number of nodes in the hidden layers equals to 4.0. The number of hidden layers was given by (number of attributes + number of classes)/2. The classic algorithms were run using the Weka 3.6 [58] tool, and the results of the PSO-based algorithms were taken from the literature. A k -fold cross-validation procedure was used to train the algorithms and estimate the prediction error, and the algorithms were run 30 times for a validation with $k = 10$ folders each. The objects of each class were distributed evenly and with stratification among the 10 folders. For *benchmarking* we

TABLE 1: Main characteristics of the databases used for performance assessment.

Databases	Objects	Attributes	Classes
Iris	150	4	3
Yeast	205	20	4
Wine	178	13	3
Glass identification	214	9	6
Haberman's survival	306	3	2
Ruspini	75	2	4
<i>E.coli</i>	336	8	5

used seven databases available in the UCI Data Repository (<http://archive.ics.uci.edu/ml/datasets.html>). The main features of these databases are listed in Table 1.

The parametric configurations used in the two algorithms proposed here have been inherited from their predecessor, the PSC. The vectors φ_1 , φ_2 , and φ_3 are random in the interval (0, 1). The inertia moment (ω) has an initial value of 0.90, with a decay of 0.95 iteratively to 0.01. The number of particles used in PSClazz was twice the number of classes present in the respective database in Table 1. The domain of the vector space was limited to [0, 1] and the velocity of the particles was also controlled and was set in the range $[-0.1, 0.1]$ to avoid particle explosion.

In its original version, the PSC stops after a fixed number of iterations. For the construction of the PSClazz, the PSC was modified such that the number of iterations required for their convergence is not defined by the user. To do that, a stopping criterion had to be proposed: the stabilization of the swarm; that is, if the average Euclidean distance between the current position and the position of the memory prototypes is less than a given threshold, then the algorithm is assumed to have converged. This stopping criterion is assessed every two iterations.

In cPSClazz, the pruning of particles occurs when they do not move during two consecutive iterations. When the similarity between the prototype and the object of greater similarity to it is greater than the affinity threshold, the prototype is cloned and the new prototype is positioned in the middle between it and the object evaluated. A suppression step was added right after the pruning step so as to minimize the number of prototypes generated.

The threshold ε depends on the dataset, for which it must be defined empirically. A number of particles added to the swarm much different from the number of classes in the database compromise the effectiveness of the algorithm. To understand the influence of ε in the cPSClazz algorithm, a sensitivity analysis of ε was performed using the databases from Table 1. The following values for ε were tested: 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, and 0.50. The results in terms of accuracy (percentage of correct classification—Pcc), number of particles (NP), and number of iterations (NI) for each value of ε tested are shown in Table 2. The results presented are the average over 30 runs, and the best results of Pcc, NP and NI on average were made in bold.

Increasing the value of ε is the same as increasing the affinity degree between the winner particle and the input

TABLE 2: Sensitivity analysis of the cPSClass algorithm in relation to ϵ for the seven datasets to be evaluated. Percentage of correct classification (Pcc), number of prototypes (NP), and number of iterations (NI) for convergence.

ϵ	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
Iris								
Pcc	88.89 $\pm$ 3.20	88.67 $\pm$ 3.11	89.78 $\pm$ 3.38	87.78 $\pm$ 4.98	87.56 $\pm$ 4.87	87.78 $\pm$ 6.57	88.22 $\pm$ 5.16	86.67 $\pm$ 7.43
NP	3.0 $\pm$ 0.0	3.07 $\pm$ 0.25	3.03 $\pm$ 0.18	3.0 $\pm$ 0.0	2.97 $\pm$ 0.18	2.97 $\pm$ 0.18	3.0 $\pm$ 0.26	2.93 $\pm$ 0.37
NI	214.63 $\pm$ 5.16	214.57 $\pm$ 5.36	212.10 $\pm$ 4.49	221.27 $\pm$ 34.06	215.50 $\pm$ 5.20	215.47 $\pm$ 5.61	218.63 $\pm$ 25.75	215.10 $\pm$ 6.61
Yeast								
Pcc	95.09 $\pm$ 1.34	99.65 $\pm$ 1.34	99.65 $\pm$ 1.34	88.22 $\pm$ 5.16	100.0 $\pm$ 0.0	99.82 $\pm$ 0.96	100.0 $\pm$ 0.0	100.0 $\pm$ 0.0
NP	3.08 $\pm$ 0.25	3.93 $\pm$ 0.25	3.93 $\pm$ 0.25	3.0 $\pm$ 0.26	4.0 $\pm$ 0.0	3.97 $\pm$ 0.18	4.0 $\pm$ 0.0	4.0 $\pm$ 0.0
NI	281.23 $\pm$ 52.78	242.40 $\pm$ 20.41	236.57 $\pm$ 9.37	219.33 $\pm$ 34.39	247.63 $\pm$ 14.70	241.60 $\pm$ 20.45	236.67 $\pm$ 8.90	239.67 $\pm$ 10.59
Wine								
Pcc	43.75 $\pm$ 0.0	57.50 $\pm$ 21.49	93.75 $\pm$ 0.0	94.17 $\pm$ 1.59	94.58 $\pm$ 2.16	95.0 $\pm$ 2.54	93.96 $\pm$ 1.14	93.96 $\pm$ 1.14
NP	1.0 $\pm$ 0.0	1.63 $\pm$ 1.0	5.93 $\pm$ 0.74	6.90 $\pm$ 0.76	6.90 $\pm$ 0.80	7.23 $\pm$ 0.86	7.10 $\pm$ 0.66	7.27 $\pm$ 0.74
NI	377.87 $\pm$ 40.95	391.80 $\pm$ 30.63	382.83 $\pm$ 39.19	343.80 $\pm$ 61.27	332.0 $\pm$ 58.82	335.87 $\pm$ 62.33	330.23 $\pm$ 56.90	323.50 $\pm$ 61.14
Glass Identification								
Pcc	52.81 $\pm$ 8.90	51.40 $\pm$ 9.44	52.46 $\pm$ 7.88	51.05 $\pm$ 7.46	50.70 $\pm$ 7.63	51.05 $\pm$ 8.97	52.63 $\pm$ 8.85	50.18 $\pm$ 9.14
NP	6.90 $\pm$ 0.40	6.90 $\pm$ 0.48	6.97 $\pm$ 0.32	6.93 $\pm$ 0.45	6.93 $\pm$ 0.52	6.87 $\pm$ 0.35	6.83 $\pm$ 0.38	6.90 $\pm$ 0.55
NI	321.97 $\pm$ 82.96	324.47 $\pm$ 82.58	308.93 $\pm$ 77.06	298.60 $\pm$ 86.12	321.0 $\pm$ 87.56	286.27 $\pm$ 79.80	304.63 $\pm$ 83.32	276.23 $\pm$ 78.83
Haberman's Survival								
Pcc	91.11 $\pm$ 3.31	92.0 $\pm$ 3.57	92.44 $\pm$ 3.15	92.0 $\pm$ 2.41	90.33 $\pm$ 3.43	91.33 $\pm$ 2.98	91.44 $\pm$ 2.72	92.22 $\pm$ 3.07
NP	8.13 $\pm$ 0.82	8.10 $\pm$ 0.61	8.13 $\pm$ 0.63	8.0 $\pm$ 0.83	8.13 $\pm$ 0.78	8.13 $\pm$ 0.73	7.97 $\pm$ 0.76	8.17 $\pm$ 0.91
NI	222.87 $\pm$ 6.14	222.10 $\pm$ 4.06	222.27 $\pm$ 5.11	221.80 $\pm$ 5.25	221.30 $\pm$ 4.91	222.10 $\pm$ 8.79	220.80 $\pm$ 7.46	221.40 $\pm$ 6.67
Ruspini								
Pcc	100.0 $\pm$ 0.0	100.0 $\pm$ 0.0	100.0 $\pm$ 0.0	100.0 $\pm$ 0.0	100.0 $\pm$ 0.0	100.0 $\pm$ 0.0	100.0 $\pm$ 0.0	100.0 $\pm$ 0.0
NP	4.0 $\pm$ 0.0	4.0 $\pm$ 0.0	4.0 $\pm$ 0.0	4.0 $\pm$ 0.0	4.0 $\pm$ 0.0	4.0 $\pm$ 0.0	4.0 $\pm$ 0.0	4.0 $\pm$ 0.0
NI	132.87 $\pm$ 65.99	151.53 $\pm$ 62.29	160.73 $\pm$ 54.09	157.90 $\pm$ 53.33	132.97 $\pm$ 62.06	120.10 $\pm$ 67.77	159.77 $\pm$ 56.46	156.50 $\pm$ 58.0
E. coli								
Pcc	82.04 $\pm$ 9.50	84.73 $\pm$ 6.45	82.26 $\pm$ 7.52	84.30 $\pm$ 7.56	81.83 $\pm$ 10.11	83.66 $\pm$ 8.96	80.97 $\pm$ 10.42	82.69 $\pm$ 9.26
NP	5.43 $\pm$ 1.04	5.73 $\pm$ 0.78	5.30 $\pm$ 0.95	5.53 $\pm$ 1.01	5.43 $\pm$ 0.82	5.60 $\pm$ 0.89	5.73 $\pm$ 0.83	5.37 $\pm$ 0.93
NI	231.07 $\pm$ 28.27	224.17 $\pm$ 10.91	236.10 $\pm$ 36.74	232.30 $\pm$ 36.84	232.40 $\pm$ 25.94	223.60 $\pm$ 16.40	232.87 $\pm$ 35.04	231.97 $\pm$ 41.43

TABLE 3: Accuracy (Pcc) and number of prototypes (NP) of cPSClazz and ScPSClazz.

Database	cPSClazz		ScPSClazz	
	Pcc	NP	Pcc	NP
Iris	95.56 $\pm$ 4.74	11.87 $\pm$ 3.25	89.78 $\pm$ 3.38	3.03 $\pm$ 0.18
Yeast	100.0 $\pm$ 0.0	23.90 $\pm$ 5.19	100.0 $\pm$ 0.0	4.0 $\pm$ 0.0
Wine	93.54 $\pm$ 2.59	15.87 $\pm$ 3.43	95.0 $\pm$ 2.54	7.23 $\pm$ 0.86
Glass	60.35 $\pm$ 7.66	22.60 $\pm$ 5.06	52.63 $\pm$ 8.85	6.83 $\pm$ 0.38
Haberman	93.67 $\pm$ 3.08	22.10 $\pm$ 3.79	92.44 $\pm$ 3.15	8.13 $\pm$ 0.63
Ruspini	100.0 $\pm$ 0.00	7.97 $\pm$ 1.13	100.0 $\pm$ 0.0	4.0 $\pm$ 0.0
<i>E.coli</i>	89.68 $\pm$ 2.86	17.97 $\pm$ 3.58	84.73 $\pm$ 6.45	5.73 $\pm$ 0.78

TABLE 4: Number of prototypes for PSClazz, cPSClazz, PPSO, MPSO, and AMPSO algorithms.

Database	PSClazz	cPSClazz	PPSO	MPSO	AMPSO
Iris	6	3.03 $\pm$ 0.18	10	16	18
Yeast	8	4.0 $\pm$ 0.0	—	—	—
Wine	6	7.23 $\pm$ 0.86	—	—	—
Glass	12	6.83 $\pm$ 0.38	10	16	19
Haberman	4	8.13 $\pm$ 0.63	—	—	—
Ruspini	8	4.0 $\pm$ 0.0	—	—	—
<i>E.coli</i>	10	5.73 $\pm$ 0.78	—	—	—

TABLE 5: Classification accuracy for PSClazz, cPSClazz, PPSO, MPSO, AMPSO, naïve Bayes, K-NN, and MLP (ANN).

Database	PSClazz	cPSClazz	PPSO	MPSO	AMPSO	Naïve Bayes	k-NN	ANN
Iris	91.78 $\pm$ 5.16	89.78 $\pm$ 3.38	90.89 $\pm$ 0.0	96.70 $\pm$ 0.0	96.89 $\pm$ 0.0	96.00 $\pm$ 0.0	96.00 $\pm$ 0.0	97.36 $\pm$ 0.0
Yeast	100.0 $\pm$ 0.0	100.0 $\pm$ 0.0	—	—	—	97.56 $\pm$ 0.0	98.05 $\pm$ 0.0	96.59 $\pm$ 0.0
Wine	93.96 $\pm$ 1.14	95.0 $\pm$ 2.54	—	—	—	96.63 $\pm$ 0.0	97.75 $\pm$ 0.0	97.19 $\pm$ 0.0
Glass	59.82 $\pm$ 6.84	52.63 $\pm$ 8.85	74.34 $\pm$ 0.0	86.27 $\pm$ 0.0	86.94 $\pm$ 0.0	49.53 $\pm$ 0.0	71.63 $\pm$ 0.0	68.37 $\pm$ 0.0
Haberman	86.11 $\pm$ 4.88	92.44 $\pm$ 3.15	—	—	—	74.51 $\pm$ 0.0	69.28 $\pm$ 0.0	74.18 $\pm$ 0.0
Ruspini	99.44 $\pm$ 3.04	100.0 $\pm$ 0.0	—	—	—	98.67 $\pm$ 0.0	100.0 $\pm$ 0.0	100.0 $\pm$ 0.0
<i>E.coli</i>	79.78 $\pm$ 10.44	84.73 $\pm$ 6.45	—	—	—	87.46 $\pm$ 0.0	84.71 $\pm$ 0.0	86.85 $\pm$ 0.0

object. Thus, when the threshold increases, the number of clones tends to increase, which can compromise the effectiveness of the algorithm. As the accuracy of the algorithm is, in principle, its most important measure of interest, the value of ϵ detached in bold in Table 2 as the most suitable one for each dataset was that with higher Pcc. It can be observed, though, that the algorithm presents some robustness in relation to ϵ , in the sense that its performance in terms of Pcc, NP, and NI changes little even with a high variation in ϵ .

To evaluate the performance of the suppression step in cPSClazz, its performance was compared with that without using it. The results are shown in Table 3 and the cPSClazz with the suppression step is identified by ScPSClazz to differentiate it from the standard cPSClazz. It can be observed that the accuracy of cPSClazz with the suppression step was worse for the Iris, Glass and *E. coli* datasets, but the number of prototypes generated was substantially smaller. By contrast, the performance of ScPSClazz was equivalent or increased even with a substantial reduction in the number of particles in the swarm.

The parametric configurations of the PPSO, MPSO and AMPSO algorithms are available in [34, 35]. The number

of prototypes of such algorithms, as well as of PSClazz and cPSClazz, is shown in Table 4. For the MLP network, the number of output neurons used was equal to the number of classes in the database (Table 1), as well as the value of k for the k -NN.

Table 5 shows the performance of PPSO, MPSO, AMPSO, PSClazz, cPSClazz algorithms and the classic algorithms from the literature when applied to the databases in Table 1. The best absolute results, on average, are made in bold in the table. The PSClazz and cPSClazz algorithms showed similar performances, on average, for the databases of Table 1. For the Yeast and Ruspini databases, the cPSClazz algorithm presented maximal accuracy, whilst no other algorithm was capable of presenting such performance. It is worth noting that the number of particles generated by the cPSClazz is greater than that used in the PSClazz for the Wine and Haberman databases, as can be observed in Table 4. Naïve Bayes, k -NN, and MLP presented a worse performance than PSClazz and cPSClazz for the Haberman databases but were competitive for the other databases. PPSO, MPSO, and AMPSO performed quite well for the Glass database, being competitive with PSClazz and cPSClazz.

A Shapiro-Wilk test [59] was used to determine whether the behavior presented by the algorithms had a normal distribution. Assuming a confidence level equals to 0.95, the test of normality revealed that the null hypothesis (H_0) should be rejected and, thus, a nonparametric test should be used to assess the statistical significance of performances. In order to determine whether the difference in performance among the evaluated algorithms is significant, we used the Friedman test [60, 61], a nonparametric method analogous to the parametric ANOVA (analysis of variance) [62]. The Friedman test is based on the ranking of the results obtained for each sample (database) j to all k algorithms. The value of the degrees of freedom is obtained by $k - 1$, being $k = 8$, so there are 7 degrees of freedom. Thus, according to the χ^2 table [63], the critical values of probability, for $\alpha = 5\%$ and $\alpha = 1\%$, are 14.07 and 18.48, respectively. As χ^2 calculated is less than the critical values, the null hypothesis H_0 is not rejected. In other words, the difference in performance between the algorithms is not statistically significant for the Haberman databases tested.

7. Conclusion

This paper presented two algorithms based on the original particle swarm optimization algorithm—PSClass and cPSClass—to solve data classification problems. The PSClass initially finds natural groups within the database, in an unsupervised way, and then adjusts the prototypes' position using an LVQ1 method, in a supervised way, in order to minimize the misclassification error. The cPSClass algorithm is similar to PSClass, except in its unsupervised phase, where it dynamically determines the number of particles in the swarm using the immune clonal selection metaphor. A parametric sensitivity analysis for cPSClass was also performed to evaluate the relation between the growth of the swarm and ϵ . For cPSClass, a suppression step was added right after the pruning step to reduce the number of prototypes generated by the algorithm.

The algorithms were applied to seven data classification problems and their performance was compared with that of algorithms well known in the literature— k -NN, MLP, and Naïve Bayes, in addition to three algorithms based on the original PSO-PPSO, MPSO, and AMPSO. It was used a k -fold cross-validation to train the algorithms and estimate the prediction error. The algorithms were run 30 times for $k = 10$ folders. The results showed that cPSClass was the best algorithm, on average, for the Haberman database, whilst AMPSO was the best for Glass, naïve Bayes was best for *E. coli*, k -NN was the best for Wine, and the MLP was the best for Iris. The PSClass and cPSClass algorithms showed similar results with each other for all the databases evaluated. However, the cPSClass has the advantage of automatically determining the number of prototypes (particles) in the swarm.

Acknowledgments

The authors thank CNPq, Fapesp, Capes, and MackPesquisa for the financial support.

References

- [1] G. E. A. P. A. Batista and M. C. Monard, "An analysis of four missing data treatment methods for supervised learning," *Applied Artificial Intelligence*, vol. 17, no. 5-6, pp. 519–533, 2003.
- [2] Y. Yang and X. Liu, "A re-examination of text categorization methods," in *Proceedings of the 22nd ACM International Conference on Research and Development in Information Retrieval*, pp. 42–49, Berkeley, Calif, USA, 1999.
- [3] X. Yi and Y. Zhang, "Privacy-preserving naive Bayes classification on distributed data via semi-trusted mixers," *Information Systems*, vol. 34, no. 3, pp. 371–380, 2009.
- [4] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufman Press, 2000.
- [5] C. Westphal and T. Blaxton, *Data Mining Solutions: Methods and Tools for Solving Real-World Problems*, Wiley Computer Publishing, New York, NY, USA, 1998.
- [6] L. M. Silva, J. Marques de Sá, and L. A. Alexandre, "Data classification with multilayer perceptrons using a generalized error function," *Neural Networks*, vol. 21, no. 9, pp. 1302–1310, 2008.
- [7] B. Gabrys and A. Bargiela, "General fuzzy min-max neural network for clustering and classification," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 769–783, 2000.
- [8] T. Kohonen, *Self-Organizing Maps*, vol. 30, Springer, Berlin, Germany, 2nd edition, 1997.
- [9] T. Kohonen, *Improved Versions of Learning Vector Quantization*, Helsinki University of Technology, Helsinki, Finland, 1990.
- [10] G. R. Lloyd, R. G. Brereton, R. Faria, and J. C. Duncan, "Learning vector quantization for multiclass classification: application to characterization of plastics," *Journal of Chemical Information and Modeling*, vol. 47, no. 4, pp. 1553–1563, 2007.
- [11] R. K. Madyastha and B. Aazhang, "Algorithm for training multilayer perceptrons for data classification and function interpolation," *IEEE Transactions on Circuits and Systems I*, vol. 41, no. 12, pp. 866–875, 1994.
- [12] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks (Perth, Australia)*, pp. 1942–1948, Piscataway, NJ, USA, December 1995.
- [13] S. C. M. Cohen and L. N. de Castro, "Data clustering with particle swarms," in *Proceedings of the World Congress on Computational Intelligence*, pp. 6256–6262, Vancouver, Canada, 2006.
- [14] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.
- [15] A. K. F. Prior and L. N. de Castro, "Um Algoritmo de Enxame Construtivo Para Agrupamento de Dados," in *Proceedings of the 18th Congresso Brasileiro de Automática*, pp. 3300–3307, Bonito, Brazil, 2010.
- [16] F. M. Burnet, *The Clonal Selection of Acquired Immunity*, Cambridge University Press, London, UK, 1959.
- [17] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.
- [18] T. Bäck, D.B. Fogel, and Z. Michalewicz, *Evolutionary Computation: Basic Algorithms and Operators*, Institute of Physics Publishing, Bristol, UK, 2000.
- [19] T. Bäck, "Evolutionary algorithms: comparison of approaches," in *Computing with Biological Metaphors*, R. Paton, Ed., pp. 227–243, Chapman & Hall, 1994.

- [20] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, Piscataway, NJ, USA, 2nd edition, 2000.
- [21] G. Beni and J. Wang, "Swarm intelligence," in *Proceedings of the 7th Annual Meeting of the Robotics Society of Japan*, pp. 425–428, 1989.
- [22] E. Bonabeau, G. Theraulaz, J. L. Deneubourg, S. Aron, and S. Camazine, "Self-organization in social insects," *Trends in Ecology and Evolution*, vol. 12, no. 5, pp. 188–193, 1997.
- [23] N. R. Franks, "Army ants: a collective intelligence," *American Scientist*, vol. 77, no. 2, pp. 138–145, 1989.
- [24] L. N. de Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer, 2002.
- [25] E. Hart and J. Timmis, "Application areas of AIS: the past, the present and the future," *Applied Soft Computing Journal*, vol. 8, no. 1, pp. 191–201, 2008.
- [26] L. N. de Castro, "Fundamentals of natural computing: an overview," *Physics of Life Reviews*, vol. 4, no. 1, pp. 1–36, 2007.
- [27] E. O. Wilson, *Sociobiology: The New Synthesis*, Belknap Press, Cambridge, Mass, USA, 1975.
- [28] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, New York, NY, USA, 1999.
- [29] J. Kennedy, R. Eberhart, and Y. Shi, *Swarm Intelligence*, Morgan Kaufman Publishers, 2001.
- [30] B. J. Zhao, "An ant colony clustering algorithm," in *Proceedings of the 6th International Conference on Machine Learning and Cybernetics (ICMLC '07)*, pp. 3933–3938, Hong Kong, August 2007.
- [31] A. Szabo, A. K. F. Prior, and L. N. De Castro, "The proposal of a velocity memoryless clustering swarm," in *6th IEEE World Congress on Computational Intelligence (WCCI '10)*, Barcelona, Spain, July 2010.
- [32] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: an overview," *Swarm Intelligence*, vol. 1, pp. 33–57, 2007.
- [33] D. W. van der Merwe and A. P. Engelbrecht, "Data clustering using particle swarm optimization," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC '03)*, pp. 215–220, Canberra, Australia, 2003.
- [34] A. Cervantes, I. Galván, and P. Isasi, "A Comparison between the Pittsburgh and Michigan Approaches for the Binary PSO Algorithm," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC '05)*, pp. 290–305, September 2005.
- [35] A. Cervantes, I. M. Galván, and P. Isasi, "AMPSO: a new particle swarm method for nearest neighborhood classification," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 39, no. 5, pp. 1082–1091, 2009.
- [36] T. Sousa, A. Silva, and A. Neves, "Particle Swarm based Data Mining Algorithms for classification tasks," *Parallel Computing*, vol. 30, no. 5–6, pp. 767–783, 2004.
- [37] N. Holden and A. A. Freitas, "A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data," in *Proceedings of IEEE Swarm Intelligence Symposium (SIS '05)*, pp. 100–107, Pasadena, Calif, USA, June 2005.
- [38] K. Y. Huang, "A hybrid particle swarm optimization approach for clustering and classification of datasets," *Knowledge-Based Systems*, vol. 24, no. 3, pp. 420–426, 2011.
- [39] Q. Shen, W. M. Shi, and W. Kong, "Hybrid particle swarm optimization and tabu search approach for selecting genes for tumor classification using gene expression data," *Computational Biology and Chemistry*, vol. 32, no. 1, pp. 52–59, 2008.
- [40] Z. Wang, X. Sun, and D. Zhang, "A PSO-based classification rule mining algorithm," in *Advanced Intelligent Computing Theories and Applications*, vol. 4682, pp. 377–384, Springer, Berlin, Germany, 2007.
- [41] S. Kalyani and K. S. Swarup, "Classifier design for static security assessment using particle swarm optimization," *Applied Soft Computing Journal*, vol. 11, no. 1, pp. 658–666, 2011.
- [42] H. Izakian and A. Abraham, "Fuzzy C-means and fuzzy swarm for fuzzy clustering problem," *Expert Systems with Applications*, vol. 38, no. 3, pp. 1835–1838, 2011.
- [43] Q. Shen, J. H. Jiang, C. X. Jiao, G. L. Shen, and R. Q. Yu, "Modified particle swarm optimization algorithm for variable selection in MLR and PLS modeling: QSAR studies of antagonism of angiotensin II antagonists," *European Journal of Pharmaceutical Sciences*, vol. 22, no. 2–3, pp. 145–152, 2004.
- [44] Q. Shen, J. H. Jiang, C. X. Jiao, S. Y. Huan, G. L. Shen, and R. Q. Yu, "Optimized partition of minimum spanning tree for piecewise modeling by particle swarm algorithm. QSAR studies of antagonism of angiotensin II antagonists," *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 6, pp. 2027–2031, 2004.
- [45] S. C. Satapathy, J. V. R. Murthy, P. V. G. D. Prasad Reddy, B. B. Misra, P. K. Dash, and G. Panda, "Particle swarm optimized multiple regression linear model for data classification," *Applied Soft Computing Journal*, vol. 9, no. 2, pp. 470–476, 2009.
- [46] W. F. Leong and G. G. Yen, "PSO-based multiobjective optimization with dynamic population size and adaptive local archives," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 38, no. 5, pp. 1270–1293, 2008.
- [47] B. Soudan and M. Saad, "An evolutionary dynamic population size PSO implementation," in *Proceedings of the 3rd International Conference on Information and Communication Technologies: From Theory to Applications (ICTTA '08)*, pp. 1–5, April 2008.
- [48] S. Sun, G. Ye, Y. Liang, Y. Liu, and Q. Pan, "Dynamic population size based particle swarm optimization," *Lecture Notes in Computer Science, Advances in Computation and Intelligence*, vol. 4683, pp. 382–392, 2007.
- [49] K. C. Tan, T. H. Lee, and E. F. Khor, "Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 6, pp. 565–588, 2001.
- [50] D. Chen and C. Zhao, "Particle swarm optimization with adaptive population size and its application," *Applied Soft Computing Journal*, vol. 9, no. 1, pp. 39–48, 2009.
- [51] G. G. Yen and H. Lu, "Dynamic population strategy assisted particle swarm optimization," in *Proceedings of IEEE International Symposium on Intelligent Control*, pp. 697–702, October 2003.
- [52] X. Cui and T. E. Potok, "Document clustering analysis based on hybrid PSO+K-means algorithm," *The Journal of Computer Science*, Special issue on Efficient heuristics for information organization, Science Publications, New York, NY, USA, 2005.
- [53] J. Dong and M. Qi, "A new algorithm for clustering based on particle swarm optimization and K-means," in *Proceedings of the International Conference on Artificial Intelligence and Computational Intelligence (AICI '09)*, vol. 4, pp. 264–268, November 2009.
- [54] F. T. Niknam and M. Nayeripour, "A new evolutionary algorithm for cluster analysis," *World Academy of Science, Engineering and Technology*, vol. 46, pp. 584–588, 2008.

- [55] S. Alam, G. Dobbie, and P. Riddle, "An evolutionary particle swarm optimization algorithm for data clustering," in *Proceedings of IEEE Swarm Intelligence Symposium (SIS '08)*, pp. 1–6, September 2008.
- [56] M. G. H. Omran, A. P. Engelbrecht, and A. Salman, "Dynamic clustering using particle swarm optimization with application in unsupervised image classification," *Transactions on Engineering, Computing and Technology*, vol. 9, pp. 199–204, 2005.
- [57] L. N. de Castro, F. J. Von Zuben, and G. A. de Deus, "The construction of a Boolean competitive neural network using ideas from immunology," *Neurocomputing*, vol. 50, pp. 51–85, 2003.
- [58] I. H. Witten and E. Frank, *Data Mining—Practical Machine Learning Tools and Techniques*, Morgan Kaufman, San Francisco, Calif, USA, 2005.
- [59] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality: complete samples," *Biometrika*, vol. 52, pp. 591–611, 1965.
- [60] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, pp. 675–701, 1937.
- [61] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.
- [62] R. A. Fisher, *Statistical Methods and Scientific Inference*, Hafner Publishing Company, New York, NY, USA, 2nd edition, 1959.
- [63] F. R. Pisani and R. Purves, *Statistics*, W.W. Norton & Company, New York, NY, USA, 3rd edition, 1998.