Pattern Recognition 46 (2013) 2028–2044

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

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

Particle swarm classification: A survey and positioning

Nabila Nouaouria [a,*], Mounir Boukadoum [a], Robert Proulx [b]

[a] Department of Computer Science, CP. 8888, Succ. Centre-ville, Montreal, QC, Canada H3C 3P8
[b] Department of Psychology, University of Quebec at Montreal, CP. 8888, Succ. Centre-ville, Montreal, QC, Canada H3C 3P8

ARTICLE INFO

Article history:
Received 12 March 2012
Received in revised form
20 December 2012
Accepted 23 December 2012
Available online 10 January 2013

Keywords:
Particle swarm optimization
Classification
High dimensional data sets
Mixed attribute data sets

ABSTRACT

This paper offers a survey of recent work on particle swarm classification (PSC), a promising offshoot of particle swarm optimization (PSO), with the goal of positioning it in the overall classification domain. The richness of the related literature shows that this new classification approach may be an efficient alternative, in addition to existing paradigms. After describing the various PSC approaches found in the literature, the paper identifies and discusses two data-related problems that may affect PSC efficiency: high-dimensional datasets and mixed-attribute data. The solutions that have been proposed in the literature for each of these issues are described including recent improvements by a novel PSC algorithm developed by the authors. Subsequently, a positioning PSC for these problems with respect to other classification approaches is made. This is accomplished by using one proprietary and five well known benchmark datasets to determine the performances of PSC algorithm and comparing the obtained results with those reported for various other classification approaches. It is concluded that PSC can be efficiently applied to classification problems with large numbers of instances, both in continuous and mixed-attribute problem description spaces. Moreover, the obtained results show that PSC may not only be applied to more demanding problem domains, but it can also be a competitive alternative to well established classification techniques.

1. Introduction

Much work has been devoted to the particle swarm optimization (PSO) metaheuristic since the mid nineties, most of it concerned with continuous function optimization. More recently, newer efforts that seek to apply PSO to more diversified problem areas have also appeared. Poli [1] lists different applications where PSO was used successfully. The applications could be divided into 26 different categories overall, after analyzing more than eleven hundred publications stored in the IEEE Xplore database. At the time, the study revealed that clustering, classification and data mining represented only 4.3% of the total production. Since then, this statistic has considerably evolved, particularly for PSO-based classification where PSO has already proven its efficiency when tested against well-know algorithms. The related literature indicates that a particle swarm optimizer is a suitable and competitive technique for addressing classification tasks, and that it can be successfully applied to demanding problem domains, especially when accurate yet comprehensible classifiers, fit for dynamic distributed environments, are required [2].

The aim of this paper is first to provide an updated survey of works using PSO for classification. Special attention is given to two major potential problems encountered in the classification domain: datasets with high dimensionality and ones with mixed-attribute data. For each problem, we analyze the solutions reported in the literature, study possible enhancement and present a performance analysis. In the remainder of this paper, Section 2 provides a brief PSC background. Section 3 presents a literature review of PSC. In Section 4, we continue our survey by focusing on the two problems mentioned above and analyze potential solutions. Section 5 presents a performance study of selected approaches, conducted on different datasets, and discusses related works. A conclusion ends the paper.

2. PSO classification background

The roots of PSO lay in ethological metaphors of computing models [2–4]. For example, the coordinated search that lets a flock of birds spot a promising food location can be modeled with simple rules of information sharing between individual birds. Such behavior inspired Kennedy and Eberhart (see [3, 5]) to develop PSO as a method for function optimization. In essence, the PSO algorithm maintains a population of particles (the

* Corresponding author. Tel.: +1 514 987 3000x4565; fax: +1 514 987 8477.
E-mail addresses: nouaouria.nabila@gmail.com (N. Nouaouria), boukadoum.mounir@uqam.ca (M. Boukadoum), proulx.robert@uqam.ca (R. Proulx).
URL: http://www.info2.uqam.ca/~boukadou (N. Nouaouria).

0031-3203/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2012.12.011

swarm), where each particle is defined by its location in a multidimensional search space (the problem space) and represents a potential solution to the optimization problem at hand. The particles start at random locations and move about the search space to look for the minimum (or maximum) of a given objective function. In the bird analogy, this function could be the quality or quantity of food at each place, and the particle swarm would search then for the place with the best or most abundant food supply. The movements of a particle depend only on its velocity and the memory of past locations where good solutions have already been found by the particle or its neighbors in the swarm. This is again in analogy to bird flocking where each individual makes decisions based on cognitive aspects (i.e., good solutions found by the particle itself) and social pressure (i.e., good solutions found by other particles). It should be noted that, unlike many deterministic methods [84] for continuous function optimization, PSO uses no gradient information in its search for solutions (As a result, there is no continuous error function requirement to compute a derivative).

Thus, in a PSO system, each particle represents a candidate solution to the optimization problem at hand and, typically, the particle's position is influenced by the best position visited by itself (its own experience) and the best position found by any particle in its neighborhood (the group experience). When the neighborhood is the entire swarm, the best neighborhood position is called the global best, and the resulting algorithm is referred to as *gbest* PSO; when smaller (local) neighborhoods are used, the algorithm is generally referred to as *lbest* PSO. The performance of each particle (i.e., how close the particle is to the global optimum) is measured by a fitness function whose form depends on the optimization problem.

More formally, a PSO algorithm is based on a swarm of M individuals or particles, each evolving in N-dimensional space with its coordinates representing a potential solution to a problem with N attributes. Its genotype consists of $2N$ parameters, the first half representing the coordinates of the particle in the problem space, and the second half the corresponding velocity components. From the evolutionary point of view, a particle moves with an adaptable velocity within the search space and retains in memory the best position it ever reached. The parameters of a particle i change from one iteration to the next as follows: Its velocity $\boldsymbol{v}_i(t+1)$ at time $t+1$ is the linear combination of its value $\boldsymbol{v}_i(t)$ at time t, the difference $\boldsymbol{b}_i(t)-\boldsymbol{x}_i(t)$ between the position of the best solution found by the particle up to time t and its current position, and the difference $\boldsymbol{b}_g(t)-\boldsymbol{x}_i(t)$ between the best position ever found by the total population and the particle's current position. We have thus:

$$\boldsymbol{v}_i(t+1) = w \times \boldsymbol{v}_i(t) + c_1 \times \boldsymbol{u}(0,1) \otimes (\boldsymbol{b}_i(t)-\boldsymbol{x}_i(t)) + c_2 \times \boldsymbol{u}(0,1) \otimes (\boldsymbol{b}_g(t)-\boldsymbol{x}_i(t)) \quad (1)$$

where bold characters denote vectors and $\otimes$ denote point-wise vector multiplication, $\boldsymbol{u}(0,1)$ is a function that returns a vector whose positions are randomly generated by a uniform distribution in [0,1], c_1 is the cognitive parameter, c_2 is the social parameter, and w is an inertia factor controlling the momentum of the particle by weighting the contributions of its previous velocity values [83]. The velocity values are normally within a range $[v_{\min}, v_{\max}]$.

An improvement to the original PSO algorithm varies the value of w during execution: Starting from a maximum $w_{\max}$, it is linearly decremented toward a minimum $w_{\min}$ as the number of iterations increases:

$$w(t) = w_{\max} - (w_{\max} - w_{\min}) \times t/T_{\max} \quad (2)$$

In the previous equation, t and $T_{\max}$ denote the current iteration and the maximum allowed number of iterations, respectively.

The position of each particle at the next step is computed by summing its current position and its velocity (assuming a unit time step):

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) \quad (3)$$

Eqs. (1)–(3) are repeated for up to $T_{\max}$ iterations, or until some predefined stopping criterion is verified. A typical convergence criterion is the achievement of a minimal error with respect to the optimal solution.

As with other stigmergic collaboration algorithms, adequate parameter tuning is important for efficient PSO performance and much work has been done to select a combination of values that work well in a wide range of problems. For instance, Clerc in [45] gave some general directives to choose the good combination. He proposed to use the following:

- Swarm size M in [20,40], with a preference for 20 particles.
- Cognitive parameter c_1 in]0,1[, with a preference for 0.7.
- Social parameter $c_2 \sim 1.5$ with a preference for 1.43.
- Maximal velocity component $v_{\max} \sim (x_{\max} - x_{\min})/2$, where $x_{\max}$ and $x_{\min}$ are, respectively, the maximum and minimum values taken by a dimension in the search space. $v_{\max}$ is used only if $c_1 > 1$ and could be different for each dimension.

In [83], Engelbrecht explained that the choice of inertia factor in the case of linear variation[1] as depicted by Eq. (2), could take $w_{\max}=0.9$ and $w_{\min}=0.4$ as initial and final values.

$T_{\max}$ the number of iterations is generally chosen as 1000 iterations.

Notice that different parameter values lead to better or worse outcomes depending on the problem at hand; the best way to tuning is to make a sensitivity analysis in the context of the problem description.

The PSO algorithm in *gbest* style can be summarized with the following pseudo-code:

Standard gbest PSO algorithm

```
(1)  begin
(2)  for each particle i
(3)     initialize position and velocity
(4)  end-for
(5)  while (not maximum number of iterations)
(6)       for each particle i
(7)          determine fitness value ψ_i
(8)          if ψ_i is better than current local best fitness
(9)             then local best fitness=ψ_i, local best=current
     particle position
(10)         if ψ_i is better than current global best fitness
(11)            then global best fitness=ψ_i and global
     best=current particle position
(12)      End-for
(13)      for each particle i
(14)         calculate particle velocity based on Eq. (1)
(15)         update particle position based on Eq. (3)
(16)      End-for
(17)      update the inertia factor based on Eq. (2)
(18) end-while
(19) end
```

The classification problem relates to finding a plausible C-partition of an input space into classes, given a C-partition of

[1] The choice of inertia factor in the static case must be in conjunction with the selection of the values for c_1 and c_2. Engelbrecht [83] reported a value of: $w > (c_1+c_2)/2-1$.

a set of training examples. As PSO is primarily an optimization tool, the problem of retrieving the C classes must be expressed in terms of retrieving C optimal particle positions corresponding to the C class centroids. To do so, the PSO algorithm performs a traversal of the search space while using a distance semantic to test the fitness of particle positions with respect to the goal. At the end, the particles with minimum fitness will define the sought centroids.

The typical PSO approach[2] to classification is a variation of basic PSO and follows the steps described in [2,6]. It can be summarized as follows: Given a database of C classes with N attributes, find C optimal real-valued coordinates in N-dimensional space, each one representing a class centroid. The idea is to start with a swarm of M particles whose coordinates are different tentative solution to the problem, and iteratively refine the positions during a training stage to find the best centroid to represent each class. During a subsequent validation stage, the found centroids are evaluated with respect to class instances in a test set to establish the recognition accuracy (or, equivalently, the percentage of classification errors). Next is a formal description of the previous steps:

Given a swarm of M particles and C class centroids to find, the ith individual in the swarm at time t is encoded by the tuple:

$$(\boldsymbol{x}^1{}_i,\ldots,\boldsymbol{x}^C{}_i,\boldsymbol{v}^1{}_i,\ldots,\boldsymbol{v}^C{}_i)_t \tag{4}$$

where $\boldsymbol{x}_i^j$ is the coordinate vector of the jth centroid as given by the current position of the ith particle in N-dimensional search space:

$$\boldsymbol{x}^j{}_i = \{x^j{}_{i1},\ldots,x^j{}_{iN}\} \tag{5}$$

Similarly, the current velocity vector of particle i with respect to the jth centroid is also made of N components:

$$\boldsymbol{v}^j{}_i = \{v^j{}_{i1},\ldots,v^j{}_{iN}\} \tag{6}$$

Thus, any individual in the swarm population is represented by a real-valued vector of $2CN$ dimensions given by Eq. (4), and each of its first C vector components defines one class centroid as found by the particle (Eq. (4)).

The fitness at time t of the ith individual in the swarm with respect to the centroid of a class c is defined as the average Euclidian distance given by:

$$\psi_i^c(t) = \frac{1}{D_{\text{Train}}}\sum_{k=1}^{D_{\text{Train}}} d(\boldsymbol{y}_k^c,\boldsymbol{x}_i^c(t)) \tag{7}$$

where $\boldsymbol{y}_k^c$ is the kth exemplar of class c in a training set of size D_{Train} and $\boldsymbol{x}_i^c(t)$ is the current position of the centroid of class c as determined by the ith particle[3], both vectors being N-dimensional.

When computing the distance, each component in the N-dimensional space is divided by its maximal range, and the sum of distance components is divided by N. With this choice, any distance will be in the interval [0.0, 1.0] and so will the fitness function.

Given the fitness function defined by Eq. (7), finding the class centroids is a typical minimization problem. During the training stage, the smaller this fitness is, the more representative the centroid pointed to by the corresponding component in the particle's position is. At each iteration, the best centroid for each class is taken as the one pointed to by the particle with the lowest fitness function value for that class. The process is repeated and the centroid positions updated until the end of the training stage. Then, the best found values for the C centroids are kept to be used for recall.

Finally, the performance of a run is computed as the percentage exemplars of each class in the testing set which are incorrectly classified by the best individual achieved in the run (in terms of fitness). Notice that the decision procedure is based on an error tolerance threshold cognitively associated to the field. The threshold is the value under which the computed fitness is considered to be satisfactory for recognition. Thus, it is the amount of error accepted in the recognition stage.

3. PSC in the literature

The classification algorithm described in the previous section is found in most works dealing with plain PSO classification[4]. There exist also works that combine PSO with other approaches, mainly neural networks. A chronological survey of recent works in each category follows:

3.1. Classification with plain PSO

In [7], PSO is extended with sequential niching methods to handle multiple minima. It combines feature-based object classification with efficient search mechanisms to visually recognize objects in an image. Each particle in the swarm is a self-contained classifier that "flies" through the solution space seeking the most "object-like" regions. The classifier swarm simultaneously finds objects in the scene, determines their size, and optimizes the classifier parameters. The approach is described as an efficient and effective search mechanism. It is also shown to be very fast and can robustly detect multiple objects in the scene.

In [8], the authors describe a self-organizing particle swarm algorithm, SOSwarm, which uses unsupervised learning. The input vectors are projected onto a lower dimensional map space, thereby producing a visual representation of the input data similar to that of the SOM (self-organizing map) artificial neural network. The particles in the map react to the input data by modifying their velocities according to the standard PSO update function, and thus organize themselves spatially within fixed neighborhoods in response to the input training vectors. SOSwarm was successfully applied to four benchmark classification problems from the UCI Machine Learning repository, namely the Wisconsin breast cancer, Pima Indians diabetes, New Thyroid and Glass, with the algorithm outperforming or equaling the best reported results on all four of the problems analyzed.

In [6], De Falco et al. evaluate PSO's efficiency for a set of classification tasks. PSO was applied to nine different datasets in the UCI database repository, and the obtained results were compared to those provided by nine classical classification algorithms available within the Waikato Environment for Knowledge Analysis[5] (WEKA) system. The alternative classification methods were as follow: Multilayer Perceptron (MLP) Radial Basis Functions (RBF) networks, KStar, Bagging, MultiBoostAB, Naïve Bayesian Tree (NBTree), Ripple Down Rule (RiDoR) and Voting Feature Interval (VFI). The parameter values for each technique were those by default in WEKA. The obtained results were that PSO had the best accuracy for three out of the nine challenged problems. Some relationships between problem size and PSO performance were hypothesized from the experimental results [6,9] to the

[2] We consider the supervised learning approach. The same model applies to the unsupervised approach; only the decision step is different.

[3] This position is refined during the learning stage (using the training dataset) to become the centroid used during the decision stage (using either a test dataset or new data).

[4] Notice that at this stage of analysis, we make no difference between classification, categorization and clustering.

[5] WEKA is a popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand.

effect that two-class problems can be suitably addressed with PSO, but no clear conclusion can be drawn for three or more class problems. In the latter case, the PSO classification accuracy tended to decrease with increasing class number; it also did so with increasing problem size. These limitations were investigated in [10] where remedial mechanisms for high-dimensional datasets were proposed.

In [11], the authors present preliminary results of two PSO-based classification approaches. They propose single and multi-surface-based data separation methods for the classification of Breast Cancer Data. In single surface separation, when a particle correctly classifies a training data item, it gets rewarded by increasing its fitness by a specific value; otherwise, when it misclassifies, it gets penalized by decreasing its fitness by a value proportional to its distance from the hyper-plane (i.e., a measure of how far on the wrong side of the surface the point lies at). The fitness values are then used to influence the location updates of the particles, and hence the position and orientation of the separating hyper-plane, in a way to drive the search towards the most optimum results. Multi-surface separation requires each particle to encode d coefficients (corresponding to space dimensions) and two constant terms (each one representing the perpendicular distance of a hyperplane from the origin) for each pair of parallel hyper-planes. The idea is based on generating an optimum set of multiple pairs of parallel separating hyper-planes to correctly classify the data into two predefined classes. For a given test record, the determination of whether the data is classified in one class or another is based on which sides of these pairs of parallel hyper-planes the data lies. Such a test is carried out in successive stages. Both separation methods produce good classification performance, with the method based on multiple separating surfaces achieving 100% classification accuracy on both the training and testing sets.

In [12], the authors propose a clustering method that is based on barebones particle swarms (BB). BB is a variant of PSO where parameter tuning is not required and in which the velocity and position update rules are replaced by a procedure that samples values from a normal distribution. The proposed algorithm finds the centroids of a user specified number of clusters, where each cluster groups a set of similar patterns. The application of the proposed clustering algorithm to the problem of unsupervised classification and segmentation of images is investigated and applied to synthetic, MRI (Magnetic Resonance Imaging) and satellite images. The experimental results show that the BB-based clustering algorithm performs very well for all measured criteria when compared to state-of-the-art clustering alternatives.

In [13–15], the authors developed a new PSO-based algorithm, called AMPSO, which can be used to find prototypes. Each particle in the swarm represents a single prototype in the solution space; the swarm evolves using modified PSO equations with both particle competition and cooperation. The experimentation part included an artificial problem and six common application problems from the UCI repository. When comparing the obtained results to other classifiers (nearest neighbor or k-NN, and linear vector quantization or LVQ), AMPSO produced competitive results in all the problems, particularly those problems where a 1-NN classifier did not perform well. AMPSO significantly outperformed the other algorithms on the Glass Identification dataset, with more than 10% improvement in accuracy on average.

In [16], the authors proposed an approach inspired by Quantum Mechanics they called Quantum-behaved particle swarm optimization (QPSO) algorithm. The particle is updated by the combination of the mean of personal best positions among the particles and a contraction-expansion coefficient. The authors apply QPSO to gene expression data clustering which can be reduced to an optimization problem. The proposed clustering algorithm partitions the N patterns of the gene expression dataset into K user-defined categories that minimize the fitness function of Total Within-Cluster Variation. A partition with high performance is thus obtained. The experimental results on four gene expression datasets show that the QPSO-based clustering algorithm was an effective and promising tool for gene–expression data analysis.

The authors in [17] applied PSO to inventory classification problems and developed a flexible classification algorithm that can be utilized as a single objective or multi-objective algorithm for different performance measures (such as cost, demand correlation or inventory turnover ratio) optimization. The algorithm can determine the best number of classification groups and classify at the same time. When performing inventory item classification, the algorithm first sorts all items according to their weight scores and then determines cut-off points along the sorting list. Items between two adjacent cut-off points are classified into the same group. To be able to handle different objectives of classification, each property is assigned a weight criterion (relevance to the objective) that is integrated to the search space. Numerical studies were conducted and the classification performance of the PSO algorithm was compared to classification by suppliers, ABC classification scheme, no grouping, and placing all items in a single group approach. The proposed algorithm outperformed the other approaches in almost all examples.

In [18], the authors addressed an image analysis problem for video surveillance using a standard PSO approach to perform unsupervised clustering of multi-class images. The overall finding of the paper after studying three separate datasets suggests that clustering using particle swarm optimization leads to better and more consistent results in terms of both cluster characteristics and subsequent recognition when compared to traditional techniques such as k-means (Especially when aiming to minimize a fitness function representing the total intra-cluster variance, as k-means implicitly does, in the case of imbalanced class datasets).

In [19], the authors designed a PSO-based clustering algorithm in which four different cluster validity indices (Euclid distance-based PBM index, the kernel function induced measure, Point Symmetry distance-based index and Manifold Distance induced index) are used to indicate the fitness of a particle. By applying the proposed algorithm to a number of artificially-synthesized and UCI data, the performances of different validity indices are compared in terms of clustering accuracy and robustness at length. The accuracy of one of the proposed algorithms for artificial data was 100%. For real data, the algorithms performed differently, with accuracies varying between 96.05% (for the Breast-cancer dataset) and 0.70% (for the Glass dataset).

In [20], the authors present a comparative analysis of k-means and PSO-based clustering performances for text datasets. Due to the high dimension of the documents database, dimensionality reduction of the input data was required during a pre-processing stage. Then, PSO clustering is applied to the reduced data using a fitness function that expresses the distance between a document and a cluster centroid. The obtained results were that PSO finds better solutions in comparison to k-means when considering the number of clusters, and the average intra and inter cluster distances. This was due to the ability of PSO to simultaneously evaluate many cluster centroids at any given time, unlike k-means.

In [21], the authors used the problem of handwritten Arabic numerals recognition to compare PSO with the Bees Algorithm (BA), Artificial Bees Colony Optimization (ABC), a multilayer perceptron neural network (MLP) and a Hybrid MLP-BA algorithm The comparative study on a variety of handwritten digits showed

the classification performance of PSO to be better than that of ABC and MLP, and worse than that of BA and MLP-BA, with the best results obtained with MLP-BA.

From what precedes, we observe the following:

- The range of application fields of PSO is very large: Text classification, videos and image analysis, inventory classification, etc.
- The classification problem is always formulated as an optimization problem, with various optimization models and fitness functions used. For instance, the fitness function could be a distance semantic, a set of multiple pairs of parallel separating hyper-planes, etc.
- Swarm configuration and behavior is not always classical (e.g., a swarm acting as a SOM).
- For most works[6], PSO performs better than other approaches when a comparison is made.

3.2. Hybrid PSO

In [22], the chaotic modeling of PSO is presented with application to image classification. The proposed algorithm is used to update the weights of neurons in Self-Organizing Feature Maps (SOFM) that tackle the underlying image classification problem. The input patterns are fully connected to all neurons via adaptable weights and, during the training process, neighboring input patterns are projected into the SOFM lattice to correspond to adjacent neurons. The performance of this modified particle swarm optimization algorithm for weight updating was found to be better than that of standard PSO when tested on binary image classes from the Corel dataset.

Omran et al. [23] proposed a new dynamic clustering approach, DCPSO (Dynamic Clustering PSO), which was applied to unsupervised image classification. DCPSO automatically determines the "optimum" number of clusters – an important feature since knowing the number of clusters in advance is often not easy – and simultaneously clusters the data with minimal user intervention. The algorithm starts by partitioning the data into a relatively large number of clusters to reduce the effects of initial conditions. Using binary particle swarm optimization the "best" number of clusters is selected. The centers of the chosen clusters are then refined via the *k*-means clustering algorithm. DCPSO was applied to natural images (including MRI and satellite images), and compared with other unsupervised clustering techniques. In general, DCPSO successfully found the optimum number of clusters on the tested images.

In [24], when training a neural network, authors observed that particles exhibited a potentially dangerous stagnation characteristic. A hybrid PSO with simulated annealing (SA) and chaos search (HPSO) is adopted to solve the problem of particles quickly collapsing. HPSO is used to train a radial basis function (RBF) neural network. The algorithm combined the desirable features of PSO, SA, and chaos while reducing their weaknesses. The application of HPSO to multiple dataset classification problems (Iris, Glass, Wine and New-thyroid) showed its effectiveness and efficiency.

In [25], the authors present a classifier based on fuzzy *C*-means (FCM) which uses a generalized FCM partition of the data optimized by PSO. The procedure consists of two phases. The first one consists of unsupervised clustering and the second one of supervised classification. The membership function parameters and the cluster center locations are optimized by PSO. Since different types of classifiers work best for different types of data, the chosen strategy is to parameterize the classifier and tailor it to individual datasets. The described FCM classifier outperformed well-established methods such as *k*-nearest neighbor, support vector machine (SVM) and Gaussian mixture classifiers on multiple UCI datasets (Iris, Breast, Glass, Wine, Pima and Ionosphere).

In [26], the authors propose a novel PSO-based classification system to improve the generalization performance of a SVM classifier by searching for the best parameter values to tune its discriminant function, and looking upstream for the best subset of features to feed the classifier. The experiments were conducted on ECG data from the Arrhythmia database in the MIT-BIH repository[7] to classify five kinds of abnormal waveforms and normal beats. The obtained results clearly confirmed the superiority of the proposed PSO–SVM classification system. Upon averaging three experiments with a different total number of training beats (250, 500, and 750, respectively), the PSO–SVM classifier had an overall accuracy of 89.72% on 40438 test beats selected from 20 patient records against 85.98%, 83.70%, and 82.34% for the SVM, *k*-NN, and RBF classifiers, respectively.

In [27], the authors propose a hybrid rough *k*-means and PSO algorithm for image classification. First, rough set theory is used to establish the lower and upper bound for data clustering in the *k*-means algorithm. Then, PSO is employed to optimize the solutions of the rough *k*-means algorithm. The combined algorithm was called the Rough *k*-means PSO algorithm. The experimental results on test images (Tool image, Tom cat image and planet image) showed that the proposed algorithm performs better than using a *k*-means only and improves the classification in the blurred and vague areas of the images.

In [28], the authors propose a novel algorithm to classify mental task signals by using PSO to train recurrent neural networks. The PSO RNN-based classification of EEG signals recorded during mental tasks (such as Complex Problem Solving, Geometric Figure Rotation, Mental Letter Composing, and Visual Counting) was investigated. The results indicated the feasibility of classifying EEG (Electroencephalography) patterns related to mental tasks. The neural network was trained and tested with mental task signals acquired from two subjects. An average classification performance of 89.9% was observed.

In [29], the authors present a new approach for the visual localization, detection, and classification of various non-stationary power signals using a variety of windowing techniques. Several non-stationary power signals are processed to extract relevant features for pattern classification. The extracted features were clustered using the fuzzy *C*-means algorithm, with further refinement of the cluster centers using PSO or a genetic algorithm. The fuzzy *C*-means algorithm is commonly used for data clustering, but it suffers from the trial-and-error choice of the initial cluster centers and the noise in the original time series. Also, the algorithm can get stuck in local minima. A hybrid fuzzy *C*-means PSO clustering technique is less sensitive to these problems. This was confirmed by the very high classification accuracy of the proposed algorithm.

In [30], the authors propose a novel particle swarm approach to evolve neural networks which aims to simultaneously optimize the architecture and the set of weights. The approach randomly generates multiple neural architectures competing with each other, while fine-tuning their architectural gaps with PSO to optimize the target model. The experiments performed on benchmark datasets from the UCI machine repository (Iris, Wine, Pima and Bupa) showed that the performance of the proposed

[6] Except cases reported in Refs. [19, 21].

[7] Available by following the link: http://ecg.mit.edu/dbinfo.html.

approach had good classification accuracy and good generalization ability.

In [31], the author introduces the use of PSO to train near-optimal decision trees. PSO is applied both in a single objective formulation (minimizing misclassification cost) and a multi-objective formulation (trading off misclassification rates across classes). The key results were the overall success of multi-objective PSO when applied to decision tree classification problems thanks to its ability to find a set of decision trees that trade-off the different misclassification error rates, and its tendency to find better single objective solutions than single-objective PSO.

In [32], the authors propose a novel method for string pattern recognition using an Evolving Spiking Neural Network (ESNN) with Quantum-inspired particle swarm optimization (QiPSO). The study is based on representing information as binary structures. The mechanism optimizes the ESNN parameters and relevant features by using the wrapper approach[8] for feature selection. String kernels are used to transform the input data to the desired input format. A kernel enables the classifier algorithm to map the original non-linear separable data into higher-dimensional space which is linearly separable. The results on Reuters string datasets show promising string classification as well as satisfactory QiPSO performance in obtaining the best combination of ESNN parameters and in identifying the most relevant features.

In [33], the authors describe a hybrid PSO–ACO (Ant Colony Optimization) algorithm to automatically classify well drilling operation stages. The basic motivation for designing the hybrid algorithm was to make PSO more effective at coping with categorical attributes, using the pheromone-based mechanism of ACO. Thanks to this, the PSO/ACO algorithm would create a classification metaheuristic that supports both nominal and continuous attributes. The stages of the drilling operation were learnt by the PSO/ACO algorithm based on a classification elaborated by a Petroleum Engineering expert; then its performance was compared to that of others classification methods: bio-inspired algorithms (artificial immune systems), decision tree learning algorithms and rule induction algorithm. In doing so, although the PSO/ACO algorithm was proposed to deal with both continuous and nominal attributes, without first converting the latter to a binary representation, only continuous attributes were considered in the experiments conducted on a mud-logging dataset. The experiments showed that, in general, the PSO/ACO algorithm is at least comparable to the rule and tree learner algorithms in terms of accuracy.

In [34], the authors are interested in selecting a relevant and discriminative combination of genes for cancer classification. A back propagation (BP) neural network is employed to construct a classifier, and PSO is used to select a discriminative combination of genes and optimize the BP classifier accordingly. Besides, sample prior information is encoded into the PSO algorithm for better performance. The proposed approach is validated on the Leukemia dataset. The experimental results showed that the method selects fewer discriminative genes while having comparable performance to the traditional classification approaches (logistic regression, neural networks, C4.5).

In [35], the authors present a new classification approach for microcalcification detection in digital mammograms using PSO and Fuzzy *C*-Means (FCM) clustering. Particle swarm optimization permits the automatic search for cluster centers in the data, using the Social-only model or Cognition-only model. The proposed classification approach is applied to a database of 322 dense mammographic images, originating from the MIAS database. The results showed that the proposed PSO–FCM approach gives better detection performance (88.50%) in comparison to conventional FCM approaches (83.4%).

In [36], a new hybrid metaheuristic learning algorithm is introduced to choose the best parameters for the PROAFTN classification method. PROAFTN is a multi-criteria decision analysis (MCDA) method which requires the values of several parameters to be determined prior to classification. These parameters include interval boundaries for the relative weights of each attribute. The proposed learning algorithm, identified as PSOPRO-RVNS because of its integration of PSO and reduced variable neighborhood search (RVNS), is used to automatically determine all PROAFTN parameters. The combination of PSO and RVNS allows to improve the exploration capabilities of PSO by setting some search points to be iteratively re-explored using RVNS. The experimental evaluations showed that PSOPRO-RVNS outperformed six well-known machine learning classifiers (Tree induction J48, Naive Bayes, Support Vector machines, multilayer perceptron, instance-based learning IBk, and rule learning PART) in a variety of problems (Breast CancerWisconsin Original, Transfusion Service Center, Heart Disease, Hepatitis, Haberman's Survival, Iris, LiverDisorders, MammographicMass, Pima Indians-Diabetes, Statlog Australian Credit Approval, Teaching Assistant Evaluation and Wine).

Here also, we observe that:

- The range of application fields of hybrid PSO is very large, with most work done in signal processing.
- An optimization problem is formulated to handle the parameters of another approach. Exceptionally, in [24,33] PSO is the object of optimization (by SA and ACO, respectively).
- For most related works, hybrid PSO performs better than other approaches it is compared to.

4. Addressed problems in the literature

Throughout our review of the literature on PSO-based classification, two problems were recurrent: the one of high dimensionality, combined or not with the large size of the database, and the problem of how to process mixed-valued data (continuous, discrete or categorical). In the remainder of this section, we analyze these problems by reviewing the approaches encountered in literature (regardless of the classification task), including PSO.

4.1. High dimensionality problem

Classification using high-dimensional features arises frequently. Fan and Fan in [38] study the impact of high dimensionality on classification. They point out that the difficulty of high dimensional classification is intrinsically caused by the existence of noise features that do not contribute to reduce the classification error. Classification using all the features can be as poor as random guessing, due to noise accumulation in estimating the population centroids in high-dimensional feature space. They demonstrate that almost all linear discriminants can perform poorly under such circumstances. Thus, it is important to select a subset of important features for high-dimensional classification.

PSC performance can also suffer from this problem. De Falco et al. [6] evaluated PSO's effectiveness at classifying data from different databases, amongst them some high-dimensional. They analyzed the total number of data instances (D), the number of classes into which it is divided (C) and the number of parameters composing each instance (N), and noted that a relationship might

[8] The wrapper feature selection approach is a variation of best-first search. When the search algorithm reaches a new node of the space, the features subset is fed to the classifier multiple times and the predictive accuracy of the classifier is estimated. The search stops when no better subsets have been found [37].

exist between PSO performance and the product ($P=D\times N$) on one part, and the number of classes C on the other part. Their conclusions were that the PSO classification accuracy tends to decrease with increasing values of C, and also with increasing values of P.

4.1.1. Solutions by search space reduction

Fodor in [39] noticed that traditional statistical methods break down for high dimensional problems partly because of the increase in the number of observations, and mostly because of the increase in the number of variables associated with each observation. High-dimensional datasets present many mathematical challenges as well as some opportunities to represent more aspects of the information, and are bound to give rise to new theoretical developments. So he proposed a survey of tools to remedy the high dimensionality problem. Fodor reviewed traditional and then current state-of-the-art dimension reduction methods published in the statistics, signal processing and machine learning literature, amongst them principal component analysis (PCA) and factor analysis (FA) which are the two most widely used linear dimension reduction methods based on second-order statistics. For datasets that are not realizations from Gaussian distributions, higher-order dimension reduction methods, using information not contained in the covariance matrix, are more appropriate. These include independent component analysis (ICA) and non-linear principal component analysis which can be considered to be a special case of it. Fodor presented some extensions of previous approaches and non-linear dimension reduction techniques such as topologically continuous maps, neural networks, vector quantization, and genetic and evolutionary algorithms.

4.1.2. Solutions by improved search

Instead of reducing the search space dimensions, the authors in [10] focus on better exploration techniques. In the literature, more sophisticated implementations of the PSO algorithm modify either the control mechanism for position updating and/or the population initialization. For instance, Liu et al. [40] propose a strategy to drive lazy particles (responsible of stagnation leading to premature convergence) and let them explore better solutions. If a particle's velocity decreases to a minimum threshold, a new velocity is assigned using a turbulence mechanism. The minimum velocity threshold of the particles is tuned adaptively by using a fuzzy logic controller. Coelho et al. [41] present a hybrid method where the PSO component uses chaotic sequences generated by a Henon map. The application of chaotic sequences instead of random sequences in PSO is a powerful strategy to diversify the population of particles and improve PSO performance by preventing premature convergence to local minima. More yet, many authors combine PSO and simulated annealing (SA) so that PSO finds a global best solution that is then refined by a local search with SA, or where SA helps to escape from local minima (see [42–44] for typical examples of use).

We present two mechanisms that have proven their efficiency in previous work [10]: confinement and wind dispersion. The confinement mechanism acts by forcing position changes to lay within an finite interval [45]. It consists of bounding the position components of a particle in such a way that, for the kth component in the N-dimensional position space, Eq. (3) becomes:

$$x_{i,k}(t+1)=\mathrm{MIN}\left(\mathrm{MAX}\left(x_{i,k}(t)+v_{i,k}(t+1),x_{\min}\right),x_{\max}\right) \tag{3a}$$

with $x_{\min}=0$ and $x_{\max}=1$.

The second mechanism, described in [22] as a chaotic search process, is wind dispersion. Wind speed and direction effects are introduced in order to model the search space's "biological atmosphere" at the time of updating particle positions. The update of the wind speed is given by the following equation:

$$\boldsymbol{vw}(t+1)=\boldsymbol{vw}(t)+v_{op}\times\boldsymbol{rand}()+v_{su}\times\boldsymbol{rand}() \tag{8}$$

where $\boldsymbol{vw}$ is the wind velocity, v_{op} is the opposing direction factor equal to -1 and v_{su} is the supporting direction factor equal to 1, and $\boldsymbol{rand}$ is a random vector generator. The wind speed has one of two effects: a particle's motion can be opposed or supported by it. The opposing effect slows down the particle in reaching the group's global best, whereas the supporting effect increases the particle velocity in reaching it. Each particle is separately updated by the wind equation. This is supported by the fact that particles are spatially separated from each other, and thus are subject to different dynamic forces from the atmosphere. When the values of the opposing and supporting wind velocities are equal, a static atmosphere is modeled. The modified position update equation for the kth dimension in the N-dimensional position space is then given by:

$$x_{i,k}(t+1)=x_{i,k}(t)+v_{i,k}(t+1)+vw_k(t+1) \tag{3b}$$

When combining this with confinement, we get:

$$x_{i,k}(t+1)=\mathrm{MIN}\left(\mathrm{MAX}\left(x_{i,k}(t)+v_{i,k}(t+1)+vw_k(t+1),x_{\min}\right),x_{\max}\right) \tag{3c}$$

The initial values of wind speed and wind direction play an important role in determining the final convergence of the particles to the optimal solution.

4.2. Mixed attribute data problem

Heterogeneous input data consisting of a mixture of numerical discrete, continuous and nominal variables are frequent in classification tasks. For example, when sorting patients into diagnostic and prognostic groups (healthy and sick) in a hospital, so as to decide whether to admit them in or treat them as outpatients, the outcome typically relies on indicators that include continuous (e.g., temperature, oxygen saturation level, etc.), nominal (e.g., presence or absence of some symptoms, patient gender, etc.) and ordinal measurements (e.g., score). Another example is the analysis of a credit application in a banking context. It usually includes classifying the application as risky or not, based on information such as age (numerical discrete), annual gain (continuous) or marital status (categorical).

When dealing with such mixed data, typical classifiers conduct a preliminary coding stage that maps the non numeric values into integers, but they face two related problems in doing so: how to establish an order relation on the transformed data that account for the actual weight of the different values or, equivalently, how to address the knowledge representation bias caused by an arbitrary coding (e.g., enumeration) that ignores the relative importance of the individual non-numeric values. In [46], an approach that avoids the previous problems by interpreting instead of plain coding is presented. The interpretation mechanism inherently reproduces the weighting semantic and is easily integrated in a metaheuristics classification approach, for instance PSO.

4.2.1. Solutions using discrete PSO

Several methods have been proposed to discretize the PSO algorithm so that it can be used to solve discrete problems, for which it is not possible to "fly" particles continuously: A survey of these methods follows:

4.2.1.1. Binary string encoding. A modification of the PSO algorithm for solving problems with binary-valued solution elements was developed by PSO's creators in [47]. The modified algorithm replaces the particle position vector update mechanism

given by Eq. (3) by a new one where the position vector components are given by:

$$x_{i,j} = \begin{cases} 1 & \text{if rand}() < S(v_{i,j}) \\ 0 & \text{other wise} \end{cases} \quad (9)$$

where $v_{i,j}$ stays as in Eq. (1) with i being the particle index and j that of the component, and $S(v_{i,j})$ is the sigmoid function. Since particles cannot fly continuously through a discrete-valued space, the significance of the velocity variable was changed to indicate the probability of the corresponding solution element assuming a value of 0 or 1. The velocity is updated in much the same way as in standard PSO, though no inertia coefficient is used here. For assigning a new particle value, the velocity term is mapped into the range (0, 1), and the particle element is randomly set with the probability of picking 1 given by $S(v_{i,j})$. This prevents the probability of it assuming a value of 0 or 1 from being too high.

Although the discrete-value modification of PSO (henceforth referred to as DPSO) was shown to be able to optimize various discrete-valued problems, it is limited to discrete problems with binary valued solution elements.

Different variants of this approach have been proposed. Al-kazemi and Mohan [48] use a technique where particles are alternatively influenced by their own best position and the best position among their neighbors. In this DPSO strategy, the velocity is updated as in standard PSO, with the difference that each element of the triple $\{w, c_1, c_2\}$ in Eq. (1) assume only the values 1 and -1, and only a given number of combinations of the different coefficients are possible. The combinations are referred to as phases of the particles and determine their directions of movement. At any given time, each particle is in one of the possible phases, and the next phase to select is determined by means of the previous phase and the number of iterations executed so far. The smallest possible non-trivial number of phases used in the DPSO method by [48] consists of two phases. In the first phase, each ith particle uses coefficients $(1, -1, 1)$ by directing the particle movement toward g_i, i.e., the best position of its social neighborhood $N(i)$. Instead, in the second phase, each ith particle uses coefficients $(1, 1, -1)$ by directing the particle movement toward its own best position b_i. A phase change occurs if no improvement of the best solution to date is obtained within a given number of iterations in the current phase.

In [49], the authors developed a DPSO algorithm that considers a larger number of coefficient combinations, referred to as quantum states of the particles, and a slightly different update equation for particle velocity inspired by the principles of quantum computing. In quantum theory, a bit is the minimum unit carrying information and it is always in a state within the range [0,1]. A quantum particle vector is defined as follows: $\boldsymbol{V}=[\boldsymbol{V}_1,\boldsymbol{V}_2,\ldots,\boldsymbol{V}_M]$, where $\boldsymbol{V}_i=[v_i^1, v_i^2 \ldots, v_i^N]$ with $0 \le v_i^j \le 1$ and $i=1, 2,\ldots,M$, $j=1, 2,\ldots,N$, N being the particle length, M being the swarm size and v_i^j denoting the probability of the jth bit of the ith particle being 0. The the rule to move from a quantum particle vector to a discrete particle vector is as follows: Assume that $\boldsymbol{X}=[\boldsymbol{X}_1, \boldsymbol{X}_2,\ldots,\boldsymbol{X}_M]$, where $\boldsymbol{X}_i=[x_i^1, x_i^2,\ldots,x_i^N]$), is the particle denotation for the practical problem, M being the swarm size. For each v_{ij}, generate a random number in the range [0,1]. If the random number is greater than v_{ij} then $x_{ij}=1$ otherwise $x_{ij}=0$. Then, the DPSO algorithm can be described by:

$$V_{localbest} = \alpha \times x_{localbest} + \beta \times (1 - x_{localbest}) \quad (10a)$$

$$V_{globalbest} = \alpha \times x_{globalbest} + \beta \times (1 - x_{globalbest}) \quad (10b)$$

$$V = w \times V + c_1 \times V_{localbest} + c_2 \times V_{globalbest} \quad (10c)$$

where $\alpha+\beta=1$, $0<\alpha, \beta<1$ are control parameters that indicate the control degree of V, $w+c_1+c_2=1$, $0<w, c_1, c_2<1$. In (10c), the first part represents the inertia of the previous probability, the second part is the "cognition" part which represents the local exploration probability; the third part is the "social" part which represents the cooperation among all quantum particles. So w, c_1 and c_2 represent the degree of belief in oneself, local exploration and global exploration, respectively.

Both methods developed in [48, 49] use the same principles as the original DPSO [47] and both are limited to discrete problems with binary-valued variables.

Pampara et al. [50] developed an indirect DPSO method by reducing a binary problem to a continuous trigonometric function having only four parameters to optimize, which allowed faster optimization of several problems. This reduction is obtained by means of angle modulation, a popular technique used in signal processing and telecommunications. The standard PSO algorithm [5] is then applied to optimize the four parameters of the continuous trigonometric function. The function is successively sampled at even intervals to produce a continuous value for each interval. If the value is positive, the bit value assigned to the corresponding interval assumes value 1, otherwise the corresponding bit value assumes value 0. The set of all generated bit values associated with the intervals represents the binary solution vector to the original binary problem. The benefit of this technique is that a larger dimensional binary space can be represented by a smaller 4-dimensional continuous space, thus allowing faster convergence of the optimization phase with respect to the other binary PSO methods in the literature. Still, the technique was only applied to binary problems.

4.2.1.2. Rounding off of particle position values. This method rounds off the continuous particle position values to the nearest integer to generate the discrete solution (see [23]). However, the method suffers from slow convergence as small velocity values (less than 0.5) are rounded to 0. So, if the velocity term is too low, the particle will not move during the corresponding iteration. Considering that complex optimization problems require thousands of iterations to complete, the occurrence of small velocity values will significantly slow down the optimization process. The authors in [51] propose to modify the PSO algorithm so that particle positions generate a continuous-valued solution. To convert the continuous-valued positions into discrete form, they propose to use the following equation (originally used to confine neural network datasets within a predefined range):

$$y = \text{round}\left(\frac{(y_{\max}-y_{\min})(r-r_{\min})}{r_{\max}-r_{\min}} + y_{\min}\right) \quad (11)$$

where y is the discrete-valued particle position, $y_{\min}$ and $y_{\max}$ are the lowest and highest values of the discrete-valued position, and $r_{\min}$ and $r_{\max}$ are the lowest and highest values of the original continuous-valued solution.

To use this equation, the continuous-valued solution should be in the range $[r_{\min}, r_{\max}]$, and the desired discrete valued solution should be defined in the range $[y_{\min}, y_{\max}]$. Thus, optimization is performed with continuous values, but discrete values are used to solve the discrete-form fitness function. After optimization, the optimal continuous-valued solution can be converted back to a discrete value using Eq. (11).

4.2.1.3. Discrete multi-valued particle swarm optimization. Multi-valued PSO (MVPSO) [52] considers variables with multiple discrete values. While the position of each particle is a one-dimensional array in continuous PSO, and a 2-dimensional array in the case of DPSO, in MVPSO, it is expressed by means of a 3-dimensional array $\boldsymbol{x}_{ijk}$, representing the probability that the ith particle in the jth iteration assumes the kth value. To evaluate the

fitness of a particle, the solution elements $\boldsymbol{x}_{ijk}$ are probabilistically generated following a sigmoid distribution, thus making the evaluation process inherently stochastic. Because the particle terms are real-valued, this representation allows velocity to be used in the same way as in standard PSO [5] where $\boldsymbol{v}_{ijk}$ represents the velocity of $\boldsymbol{x}_{ijk}$. Therefore, it is still possible to update the velocity v_{ijk} by means of the classical equation.

4.2.1.4. Jumping particle swarm optimization (JPSO). JPSO is a new DPSO proposed in [53], and extended in [54]. The approach does not consider any velocity since, from the lack of continuity of movement in a discrete space, the notion of velocity has less meaning; however the attraction by the best positions is kept. JPSO considers a swarm of particles whose positions evolve in the solution space, jumping from one solution to another. For each iteration, each particle has a random behavior or jumps to a solution in a manner guided by the effect of an attractor. The algorithm considers three attractors for the movement of a particle i: its own best position to date (b_i), the best position of its social neighborhood (g_i), interpreted as the best position obtained within the swarm in the current iteration, and the best position to date obtained by all the particles, which is called the global best position (g^*). A jump approaching an attractor consists of changing a feature of the current solution by a feature of the attractor. Each particle is further allowed to behave randomly by performing random jumps. These consist of randomly selecting a feature of the solution and changing its value. A natural inspiration for this process is found in frogs jumping from a lily pad to another in a pool, hence the name Jumping particle swarm optimization. The authors show that JPSO is able to obtain good approximate solutions for large size data sets.

4.2.2. Attribute type-based position interpretation for mixed-attribute PSO

The previous studies were all concerned with modeling PSO as a continuous or discrete-valued PSO problem, but as mentioned before, this creates potential problems when facing nominal attributes (defined here as attributes that take a finite number of non-numeric values, categorical or symbolic) and mapping their non numeric values into integers as usually done. Such cases create the problems of how to establish an order relation on the transformed data and how to address the bias caused by a flat representation, with the result of losing the real relative importance of the individual non-numeric values after the coding process.

In [55] a PSO algorithm with an attribute type-based position interpretation mechanism was proposed for retrieving cases having a mixed-attribute description from a case base. It was used again in Refs. [46, 56] for a classification task tested on various datasets. The approach avoids the previous problems by interpreting instead of coding, and the interpretation mechanism inherently reproduces the weighting semantics and is easily integrated in a metaheuristic-based classification approach as it is uniformly applicable to classification tasks with nominal, continuous and discrete input data. The main idea is to start from a continuous PSO algorithm and use mechanisms for the interpretation of particle positions in the right way. Two spaces are considered: a search space in which particles evolve with continuous coordinates as in standard PSO and a description space reflecting reality, where the input vectors are expressed with continuous, numerical discrete or nominal components. It follows that while a particle evolves along continuous axes in the particle space, its positions are interpreted in terms of descriptors (attributes) of miscellaneous natures in the description space. A semantic mapping between the two spaces allows moving from one to the other. It is ensured by a set of interpretation mechanisms that allow the continuous values in the first space to have corresponding continuous, discrete or nominal values in the second. Essentially, one of the following measures is called upon between the search and description spaces:

- Identity or affine transformation for continuous attributes. As a result, the continuous particle coordinates map to continuous equivalents in the description space as done in standard PSO.
- Rounding for integer attributes as often done in discrete PSO. This method rounds off the continuous particle coordinates to the nearest integer to generate the discrete solution in description space.
- Frequency substitution for nominal attributes. The corresponding particle coordinates in the search space are interpreted as the attribute frequencies in the description space.

Thanks to the interpretation mechanisms, the PSO algorithm keeps functioning as a continuous model; only the interpretation of particle positions changes. The changes occur prior to evaluating the fitness function which still expresses the semantics of the description space. Velocity, position and inertia keep evolving in the continuous search space, and only the fitness function is evaluated with the interpreted values of the mixed attributes in the description space. So, the values of $\boldsymbol{y}_k^c$ and $\boldsymbol{x}_i^c$ in Eq. (7) refer to attribute values in the description space, although their search space equivalents are used. In other terms, we search in 'search space' but compare in 'description space'.

5. Classification performance and PSC positioning

In the previous section, we discussed the potential problems of high dimensionality and how to process mixed-valued data. In this section, we continue our analysis in an empirical experimental classification context which will also enable us to situate PSC with respect to conventional classification techniques in terms of accuracy. Thus, we performed a positioning study which was done from a data use perspective since we cannot say how a classification approach performs in the absolute. We used a context composed of one proprietary dataset and 5 datasets from the UCI Repository [57]. These were the FLUO (proprietary), LET and OPTIDIGIT datasets for testing high dimensionality, and the Adult, CMC and Abalone datasets for testing mixed-attribute data.

5.1. Results for high dimensionality problem

5.1.1. FLUO dataset

The FLUO data set was used in [58]. The authors used a multi-wavelength sensor to acquire fluorescence data from various organic substances at different concentrations. Then, they trained an artificial neural network (ANN) with the collected data and used the ANN to classify new data collected by the sensor. The aim of classification was to predict target classes predefined to correspond to specific substance-concentration pairs, for given input patterns. The FLUO dataset has the following characteristics:

- Number of instances for the training set = 2103.
- Number of instances for the test set = 1051.
- Input space dimension = 64.
- Number of classes = 19.

The ANN architecture in [58] is based on a multilayer perceptron (MLP) with two hidden layers and one output neuron, trained with the resilient backprobagation (RPROP) algorithm.

The system was tested to identify four fluorescent organic compounds at different concentrations, chosen in view of their excitation and emission wavelengths falling within the range of the sensor's front end. The obtained results showed an excellent performance of the system at identifying and quantifying the compounds at different concentrations, with over 95% recognition accuracy (Table 1).

In [10], a classical PSO was applied first. The obtained recognition accuracy was very bad. These results tend to corroborate the conclusions of [6] to the effect that PSO is less performing for data sets with high dimensions. However, when a confinement mechanism was used to update particle positions, the accuracy jumped to 100% and a computation cost of 1.05 min. This jump could be explained by the fact that the input data from the sensing device were normalized. Thus, the particles should have evolved within a bounded search space with position components in the range [0,1]. In reality, the position components took very large values, and it was the mechanism of interval confinement [45] that solved the problem. When combined to wind dispersion, the same results as with confinement mechanism were observed.

5.1.2. LET dataset

The LET (for Letter recognition) dataset in the UCI Repository [57] relates to the problem of classifying typed upper case letters of the Latin alphabet based on a number of statistical properties of their pixel images. LET consists of 20000 unique letter images obtained by randomly distorting pixel images of the 26 uppercase letters from 20 different commercial fonts. The parent fonts represent a full range of character types including script, italic, serif, and Gothic. The features of each of the 20 000 characters are summarized in terms of 16 primitive numerical attributes. The LET database has the following characteristics:

- Number of instances for the training set=16 000.
- Number of space dimension=16.
- Number of classes=26.
- Number of instances in the test set=4000.

The first use of LET was in [59] where the authors describe an adaptive supervised classifier system which creates lists of fixed length condition-action rules (i.e., classification rules) that are applied in parallel to "messages" representing the presence or absence of specific features in the current environment. Classifier systems typically possess three major components: a performance algorithm that compares rules with messages to determine which rule(s) should be activated, a reinforcement algorithm that modifies the strength of each rule, based on its "fitness" in the current environment, and a rule-creation algorithm that generalizes exemplars or combines current rules to produce new ones. The research for this article investigated the ability to learn to correctly guess the letter categories associated with vectors of 16 integer attributes extracted from images of letters. The best accuracy obtained was slightly over 80%.

Table 1
Reported classification performance on FLUO data set.

Approach	Reference	Accuracy (%)	Learning time
MLP	[58]	95	N/A
PSO (classical)	[10]	64.29	8.38 min
PSO with confinement	[10]	100	1.05 min
PSO with wind dispersion and confinement	[10]	100	1.06 min

Notes: N/A=Not available; PSO results based on an error tolerance of ± 0.005 for the fitness function.

More recently, the authors in [60] developed a new algorithm for the computation of piecewise-linear boundaries of finite point sets. Their algorithm consists of two main stages. In the first one, they compute a hyperbox that approximates each class. Then, the obtained hyperboxes allow identification of so-called indeterminate regions, where data points from different classes are mixed. Data points that belong to only one hyperbox are called classified points, and they are removed from further examination. In most cases, the use of hyperboxes allows to significantly reduce the number of data points and, consequently, the computational effort. After implementation of the algorithm, called Maxmin, the authors tested it on LET and used a number of classifiers from the WEKA suite for comparison. The authors chose a representative of each type of classifier in WEKA: Naive Bayes (with kernel), Instance-based algorithm IBk (with $k=5$), Logistic Regression based classifier Logistic, Multi-Layer Perceptron (MLP), support vector machine classifiers, Linear LibSVM (LIBSVM (LIN)) and decision tree classifier J48 (which is an implementation of the C4.5 algorithm) and the rule based classifier PART. In addition, they also tested Poly, a classifier based on polyhedral separability [61]. Since the number of hyperplanes in polyhedral separability is not known a priori, they ran the algorithm with 2 to 5 hyperplanes and reported the best results on the test set. The best classification result was an accuracy of 94.96%, obtained with the Instance-Based-Algorithm (see Table 2).

In [56], the application of classical PSO led to a recognition accuracy of 69% for a computation cost of 42.21 min. When the particle update rule used confinement, the recognition accuracy decreased to 64.27% with a computation cost of 44.22 min, showing that interval confinement alone does not perform well all the time. On the other hand, the application of both wind dispersion and confinement led to the best classification results of all classifiers that we surveyed, with a value of 95.20% and a learning time of 23.40 min (see Table 2).

5.1.3. OPTIDIGIT dataset

This dataset contains handwritten digits and was used in [62] for classification performance testing. The authors used preprocessing programs made available by NIST (National Institute of Standards and Technology) to extract normalized bitmaps of handwritten digits from a preprinted form. The 32 by 32 normalized bitmaps were low-pass filtered and undersampled to get 8 by 8 matrices where each element is an integer in the range 0 to 16. 44 people filled in forms. The authors propose a multistage recognition system that consists of cascading a linear parametric

Table 2
Reported classification performance for LET data set.

Approach	Reference	Accuracy (%)	Learning time
Adaptative classifier systems	[59]	80.00	N/A
NB	[60]	74.12	N/A
IBk	[60]	94.96	N/A
Logistic	[60]	77.40	N/A
MLP	[60]	83.20	N/A
LIBSVM	[60]	82.40	N/A
J48	[60]	87.70	N/A
Maxmin	[60]	91.82	N/A
PART	[60]	87.32	N/A
Poly	[61]	88.68	N/A
PSO (classical)	[56]	69	42.21 min
PSO with confinement	[56]	64.27	44.22 min
PSO with wind dispersion and confinement	[56]	95.20	23.40 min

Notes: N/A=Not available; PSO results based on an error tolerance of ± 0.005 for the fitness function.

model and a k-nearest neighbor (k-NN) nonparametric classifier The linear model learns a 'rule' and the k-NN learns the 'exceptions' rejected by the 'rule'. Instead of finding a complex rule that explains all the cases, the idea is to have a simpler linear model that explains a large percentage of the cases, keeping a list of the cases not covered by the rule as exceptions. Thus, inputs rejected by the first stage are handled by the second stage which uses costlier features or decision making mechanisms. The OPTIDIGIT database is characterized by:

- Number of instances for the training set=3823.
- Number of instances for the test set=1797.
- Input space dimension=64.
- Number of classes=10.

The best recognition accuracy for the test set using the approach of [62] was 98% when using $k=1$ and a Euclidean distance metric for the k-NN classifier. The performance of the first classifier was given in term of the percentage of patterns stored to be treated by the second classifier, and it was between 3% and 7% (Table 3).

In [63], the authors studied how evolutionary algorithms scale for larger datasets, and experimented a '1+1' evolutionary strategy (one parent and one mutant) and a simple genetic algorithm on OPTIDIGIT. The larger size of the training set could cause fitness evaluations to be prohibitively expensive, and therefore they sought to obtain faster approximate evaluations by sampling the training set. The best accuracy they obtained on the entire data was $90.2\% \pm 1.1$ with a time consumption of 144.2 s.

Table 3
Reported classification performance for OPTIDIGIT data set.

Approach	Reference	Accuracy (%)	Learning time
k-NN	[62]	98.00	N/A
EA	[63]	90.2 ± 1.1	144.2 s
C4.5	[64]	79.58	N/A
C4.5 Boosted	[64]	95.10	N/A
PSO (classical)	[10]	98.43	32.88 min
PSO with confinement	[10]	96.42	35.09 min
PSO with wind dispersion and confinement	[10]	99.44	9.36 min

Notes: N/A=Not available; PSO results based on an error tolerance of ± 0.005 for the fitness function.

In [64] the authors experimented with different techniques to boost the performance of the C4.5 algorithm, the best result they obtained was of 95.10% instead of 79.58% for the non boosted C4.5 algorithm.

In [10], classical PSO gives a recognition accuracy of 98.43% for a computation cost of 32.88 min, and the addition of the confinement mechanism yielded 96.42% of recognition accuracy for a computation cost of 35.09 min. As with the LET dataset, the combination of confinement with wind dispersion led to the best results with a recognition accuracy of 99.44% and a learning time of 9.36 min.

5.2. Results for mixed attribute data sets

5.2.1. Adult dataset

In addition to a high dimensionality characteristic, the Adult dataset has mixed attribute descriptions since the 14 attributes of each of its 48 842 records are a mix of continuous and discrete (numerical and nominal) values. The dataset was originally developed for the prediction of whether a person's yearly income is higher than 50 k$. The authors in [57] also report the prediction error rates of different known classification algorithms.

Over fifty papers exist in the literature in relation to the Adult dataset. We focus on the ones that use supervised learning for the classification task. These use mainly decision tree (C4.5), naïve Bayesian (NB and NB tree), Artificial Neural Networks (ANN), k-Nearest Neighbor (k-NN) and support vector machine (SVM) approaches.

Table 4 provides the best accuracies achieved by the different works we surveyed. Unfortunately, it is difficult to make precise comparisons with these results since not all the works use the entire dataset, and many either ignore instances with missing values (7% of the total number of instances), utilize a reduced set of instances, or only consider a subset of the attributes.

In [65], the authors present the Naive Bayes Tree algorithm (NBTree), a hybridation of Naïve Bayes and decision trees classifiers. NBTree is similar to the classical recursive partitioning schemes, except that the leaf nodes are Naïve_Bayes categories instead of nodes predicting a single class. In [66], the authors present a Naïve or Simple Bayes Classifier (SBC) which assigns to each test example a score between 0 and 1, interpreted as a class membership probability estimate. Simple Bayes has been used as classifier for many years since it is easy to construct and the classification process can be efficient for large datasets. However, the approach relies on the assumption that the attributes of

Table 4
Reported classification performance for adult dataset.

Approach	Reference	Data size (train/test)	Accuracy	Training time
NBTree	[65]	30162/15060	85.90 ± 0.28	N/A
SBC	[66]	32561/16281	84.18 ± 0.29	N/A
C4.5	[66]	32561/16281	85.97 ± 0.27	N/A
TAN	[67]	32561/16281 (13 attributes)	86.01 ± 0.27	131 s
BAN	[67]	32561/16281 (13 attributes)	85.82 ± 0.27	536 s
GBN	[67]	32561/16281 (13 attributes)	86.11 ± 0.27	515 s
SVM	[68]	400 training instances from non-missing value data	84.70 ± 0.30	Few minutes
SVM	[69]	32561/16281	85.02	1065.9 s
ANN	[70]	48842 (*)	80.87	N/A
k-NN	[70]	48842 (*)	80.70	N/A
PSO(classical)	[46]	32561/16281	99.83%	9.78 h
PSO with wind dispersion and confinement	[46]	32561/16281	100%	6.48 h
PSO with wind dispersion and confinement	[46]	48842 (*)	99.56%	17.74 h
PSO with wind dispersion and confinement	[46]	4883/4883	91.77	1874.2 s

Notes: N/A=Not available; PSO results all based on an error tolerance of ± 0.003 for the fitness function, except the last one where the error tolerance is ± 0.005.

* Ten-fold cross validation.

examples are independent given the class of the examples, which can be a problematic. The authors in [66] focused on the robustness of the approach to obvious violation of the independence assumption and on various techniques to process unknown values. They also report results with the inductive learning C4.5 algorithm.

In [67], the authors empirically evaluate algorithms for learning three types of Bayesian network (BN) classifiers and discuss how these methods address the overfitting problem and provide a natural method for feature subset selection. The BN were: tree augmented Naïve-Bayes (TAN), BN augmented Naïve-Bayes (BAN) and general BNs (GBN). The TAN classifier extends Naïve-Bayes by allowing attributes to form a tree while the BAN classifier extends TAN by allowing attributes to form an arbitrary graph instead of a tree. GBN is another type of unrestricted BN classifier. A common feature of TAN and BAN is that the class node is treated as a special node (the parent of all the features). On the other hand, GBN treats the class nodes as ordinary nodes and is not necessarily a parent of all the feature nodes.

In [68], the authors propose an approach that operates on a reduced set and keeps the number of model parameters small in order to remedy the problem of kernel-based methods, the excessive time and memory requirements when applied to large datasets. The kernel methods' mapping of data into higher dimensional spaces for better performance at classification and regression tasks becomes a bottleneck when this dimensionality is important, since the corresponding training methods scale with polynomial complexity for large instance problems. In their paper, the authors present a variant of least square support vector machine (LS-SVM), which operates in primal space and has two important advantages: a small number of regression coefficients (which allows a fast training) and a sparse kernel expansion (which allows fast evaluation). For the Adult dataset, the LS-SVM was trained and cross-validated with reduced datasets of size $m=400$, thus ignoring information that may have been conveyed only by the entire training set.

Also addressing the problem of excessive time and memory requirements when applying kernel-based methods to large datasets, the authors in [69] show that decomposition methods based on alpha seeding techniques are extremely useful for solving a sequence of linear SVMs with more data than attributes. The decomposition method is an iterative procedure where, for each iteration, the index set of the variables is separated in two. The variables corresponding to a non working set are fixed while a sub-problem on variables corresponding to working set is minimized. The term alpha seeding is used to refer to any method which provides initial estimates of the alpha values for the optimization problem (instead of using the default of all zero alphas that usual SVM methods use). The alpha seeding approach performs so well that its total number of iterations is much less than solving one single linear SVM with the original decomposition implementation.

In [70], the authors address the robustness of commonly used classifiers to changing environments. They simulate the changing environments by reducing the influence on the class of the most significant attributes. Based on their analysis, *k*-Nearest Neighbor (*k*-NN) and artificial neural networks (ANN) are the most robust learners.

In [46], the application of classical PSO gives a recognition accuracy of 99.83% for a learning time of 9.78 h. When applying wind dispersion combined with confinement, the approach reaches perfect accuracy and a lower learning time of 6.48 h (in a 2/3 and 1/3 validation). The accuracy slightly diminishes to 99.56% with a mean learning time of 17.74 h in a ten-fold cross validation. In all cases, PSO broadly wins in accuracy, but loses in time consumption in comparison to the other classification methods. It should be noted, however, that the training time is intimately linked to the size of the training set. For instance, the last row of Table 4 shows that when only one tenth of the Adult database is used for training, the time consumption decreases similarly.

5.2.2. CMC data set

The contraceptive method choice (CMC) dataset was developed for the prediction of women contraceptive choices. The data are taken from the 1987 National Indonesia Contraceptive Prevalence Survey. The dataset has 1473 instances with 9 inputs (2 continuous, 4 categorical and 3 binary). The problem is to predict the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socio-economic characteristics (e.g., wife's age, religion).

Numerous papers also exist in the literature in relation to the CMC data set. They use mainly statistical approaches (Nearest Neighbors, Polyclass), decision trees (C4.5 enhanced, QUEST) and more recently, population-based approaches and hybrid approaches. Table 5 provides the best accuracies achieved by the different works that we surveyed. All the works used a 10-fold cross validation on the CMC data set.

One of the most exhaustive studies was due to Lim et al. [71]. It is a comparative study of 33 classification algorithms (22 decision-tree, 9 statistical and 2 neural network algorithms) on 32 datasets. Their performance was expressed in terms of classification accuracy, training time, and number of leaves in the case of trees. Lim and coworkers also ranked datasets by their ease of classification, with the finding that CMC was among the most difficult to classify. For that reason, only 3 of the 33 approaches were tested on CMC, namely:

- From the statistical family, they used the classical Nearest Neighbor and POLYCLASS [72]. POLYCLASS fits polytomous logistic regression model using linear splines and their tensor products. It provides estimates for conditional class probabilities which can then be used to predict class labels.
- From the decision tree classifier family, they used QUEST (Quick, Unbiased, Efficient, Statistical Tree). QUEST has negligible variable selection bias, computational simplicity, it includes pruning as an option, and yields binary splits. The reason for binary splits is that the QUEST trees may then be easily compared with exhaustive search trees in terms of stability of the splits and number of nodes [73]. We report the results of two variants: QU0 and QU1 from [71].

In [74] the authors describe a hybrid inductive machine learning algorithm called CLIP4. The algorithm first partitions the data into subsets using a tree structure and then generates production rules from the subsets stored at the leaf nodes. The algorithm allows the generation of rules that involve inequalities. It works with data that have a large number of examples and attributes, can cope with noisy data, and can use numerical, nominal, continuous, and missing-value attributes. It handles nominal data by automatic front-end encoding into numerical values. CLIP4 uses a Genetic Algorithm (GA) to improve the accuracy of the generated rules. CLIP4's genetic module works by exploiting a single loop through a number of evolving populations. The loop consists of establishing the initial population and subsequently performing selection of the new population from the old population, alteration and evaluation of the new population, and substitution of the old one with the new population. CLIP4 uses GA to enhance the partitioning of the data and, possibly, to achieve more general leaf node subsets.

Table 5
Reported classification performance for CMC dataset.

Approach	Reference	Accuracy (%)	Training time
Nearest neighbors	[71]	71.9	N/A
POL	[71]	80.5	3.2 h
Decision tree (QUEST)	[71]	77.90	NA
CLIP4	[74]	47.00	46 s
CM (soft)	[75]	57.50	1000 s (termination criterion)
CM (hard)	[75]	56.00	1000 s (termination criterion)
PM (soft)	[75]	65	1000 s (termination criterion)
PM (hard)	[75]	63.9	1000 s (termination criterion)
DM (soft)	[75]	57.5	1000 s (termination criterion)
DM (hard)	[75]	56.2	1000 s (termination criterion)
LM (soft)	[75]	67.5	1000 s (termination criterion)
LM (hard)	[75]	67	1000 s (termination criteria)
BM (soft)	[75]	69	1000 s (termination criteria)
BM (hard)	[75]	69	1000 s (termination criteria)
GAssist	[76]	54.90 ± 4.11 (*)	N/A
XCS	[76]	53.59 ± 3.56 (*)	N/A
GS	[77]	69.79	N/A
G	[77]	67.35	N/A
SS	[77]	64.71	N/A
BG	[77]	63.54	N/A
PSO with wind dispersion and confinement	[56]	97.27	84.94 min

Notes: N/A=Not available; PSO result based on an error tolerance of ± 0.027 for the fitness function.

* Best results amongst different configurations.

In [75], five distinct algorithms are defined for the learning task: (1) The Connectionist Model (CM) where the learning is achieved by a single Multi Layer Perceptron (MLP) trained by the Retro-propagation (RPROP) algorithm; (2) The Population of Connectionist Models (PM) where a set of 20 MLP individuals evolve by only using the RPROP learning algorithm (no reproduction or selection procedure is applied); (3) The Darwinian Model (DM) where the learning is accomplished by Evolutionary Programming (EP) with a population of 20 real-valued chromosomes is evolving, each coding the weights of a MLP; (4) The Lamarckian Model (LM) which combines both lifetime learning and evolutionary approaches; (5) The Baldwinian Model (BM) which is similar to LM, except that lifetime learning is only used to improve the fitness of the individuals, and the new weights are not encoded back into the genome. This means that, in the process of reproduction, the offspring does not inherit the acquired genetic information from their ancestors.

The experimentations used two types of environments: soft changing or concept drift (one pattern will change at each second) when changes occur gradually; and hard changing or concept shift (several patterns will be commuted over a wider period) when changes occur abruptly. In Table 5, we report 10 results from [75], corresponding to the five approaches (CM, PM, DM, LM and BM) with the two environments (Soft and Hard).

In [76], the authors compare the performance of the GAssist system with that of the XCS system on several data mining problems. GAssist is a genetic-based machine learning system. It applies a near-standard GA that evolves individuals that represent complete problem solutions. An individual consists of an ordered, variable-length rule set. Bloat control is achieved by a combination of a fitness function based on the minimum description length (MDL) principle and a rule deletion operator. The knowledge representation used for real-valued attributes is called adaptive discretization intervals rule representation (ADI). This representation uses the semantics of conjunctive normal form predicates, but applies non-static intervals formed by joining several neighbor discretization intervals. These intervals can evolve through the learning process splitting or merging among them potentially using several discretizers at the same time. The system also uses a windowing scheme called ILAS (incremental learning with alternating strata). This scheme stratifies the training set into subsets of equal size and approximately uniform class distribution. Each GA iteration uses a different stratum to perform its fitness computation, using a round-robin policy. This method was shown to introduce an additional implicit generalization pressure to GAssist. The XCS classifier system evolves online a set of condition-action rules, that is, a population of classifiers. In contradistinction to GAssist, the population as a whole represents the problem solution in XCS. XCS is featured by: (1) Rule fitness is derived from rule accuracy instead of rule reward prediction. (2) GA selection is applied in the subsets of currently active classifiers resulting in an implicit pressure towards more general rules.

In [77], the authors extends previous work on skewing, an approach to problematic functions in decision tree induction. The previous algorithms were only applicable to functions of binary variables. In [77], skewing is extended to directly handle functions of continuous and nominal variables. Experiments are presented to compare the performance of a tree learner using the Information Gain (G) criterion and the Generalized Skewing (GS) criterion for selecting split variables. The base tree learner is comparable to C4.5 with the "subset-splitting" option. The authors also experiment sequential skewing (SS) and Gain (BG) on the binarized versions. Their results indicate that the GS algorithm almost always outperforms an Information Gain-based decision tree learner. We report in Table 5 their results for CMC data set.

The results obtained in [56] by PSO with wind dispersion and confinement were the best amongst all with an accuracy of 97.27% in a training time of 84.97 min.

5.2.3. *Abalone data set*

The Abalone dataset contains physical measurements of the abalone shellfish. The dataset contains 4177 samples with 9 attributes each (1 categorical and 8 numeric), divided into 29 classes. The age of an abalone can be determined by cutting the shell through the cone, staining it, and counting the number of rings with a microscope. In practice, easier to obtain measurements are used to predict the age (Length, Diameter, Height, Whole weight, Shucked weight, Viscera weight, Shell weight).

Multiple papers exist in the literature in relation to the Abalone data set. We focused on the ones that use supervised learning for classification to compare their results with ours. These use mainly decision tree (C4.5), naïve Bayesian (NB and NB tree) and support vector machine (SVM) approaches.

Table 6 provides the best accuracies achieved by the different works that we surveyed. Unfortunately, no training times accompanied the results that were reported in these works.

In [78], the authors describe a new instance-based regression method that uses feature projections called regression by partitioning feature projections (RPFP). Feature projection approaches store the training instances as their projected values on each feature dimension separately. These projections can be generalized into feature intervals. In predicting the target value of a query instance, each feature makes a separate prediction using only the value of the query instance for that feature, then all the feature predictions are combined to make the final prediction. Feature projection-based techniques have been applied to many classification problems successfully. The main advantage of feature projection-based classification methods is their short classification time. The concept representation in the form of feature intervals can be transformed into decision rules easily. They are also robust to irrelevant features and missing values. However, the main shortcoming of feature projection-based methods is that they ignore the interactions between features. The RPFP method is adaptive and robust to irrelevant features. It is not a simple first order projection-based technique; it uses projections and also handles interactions between input variables. The authors also provided a comparative study with other important approaches in the literature. The approaches considered are instance based regression, *k*-NN, locally weighted regression, rule-based regression (RULE), partitioning algorithms that induce decision trees (DART) and multivariate adaptive regression splines (MARS). A detailed overview of these regression techniques is given in [79].3

In [80], three learning algorithms are used: C4.5, a decision tree learning algorithm; NB, a re-implementation of a Naive Bayesian classifier; and IB1, a variant of a lazy learning algorithm which employs the *p*-nearest-neighbor method using a modified value-difference metric for nominal and binary attributes. Their results were compared to MLR, which is an adaptation of a least-squares linear regression algorithm. The idea is that any classification problem with real-valued attributes can be transformed into a multi-response regression problem. If the original classification problem has I classes, it is converted into I separate regression problems, where the problem for class l has instances with responses equal to one when they have class l and zero otherwise. For all those algorithms (C4.5, NB, IB and MLR), a stacked generalization is used to enhance accuracy. Stacked generalization is a way of combining multiple models that have been learned for a classification task, stacking the first step to collect the output of each model into a new set of data which will be used for a second learning step.

The best classification performances obtained in the literature for the Abalone dataset were those of [81] where a generalized Gaussian process (GP) (for details see [82]) framework for regression is presented. It learns a nonlinear transformation of the GP outputs. The learning algorithm chooses a nonlinear transformation such that transformed data is well-modeled by a GP. This can be seen as including a preprocessing transformation as integral part of the probabilistic modeling problem, rather than as an ad-hoc step. They show how such a transformation or 'warping' of the observation space can be made entirely automatically, fully encompassed into the probabilistic framework of the GP. The warped GP makes a transformation from a latent space to the observation, such that the data is best modeled by a GP in the latent space. It can also be viewed as a generalization of the GP, since in observation space it is a non-Gaussian process, with non-Gaussian and asymmetric noise in general.

The results obtained in [56] by PSO with wind dispersion and confinement ranked in second position with an accuracy of 94.41. It should be noted, however, that the first ranked approach, namely GP variants, used training/test sets sizes of 1000/3177 with no cross validation.

Table 6
Reported classification performance for Abalone dataset.

Approach	Reference	Accuracy (%)
KNN	[78]	33.90
RULE	[78]	10.10
MARS	[78]	31.70
DART	[78]	32.20
RPFP	[78]	32.50
C4.5	[80]	61.50 (*)
NB	[80]	61.50 (*)
IB1	[80]	61.80 (*)
MLR	[80]	61.90 (*)
GP	[81]	95.21 (**)
GP+log	[81]	95.38 (**)
Warped GP	[81]	95.37 (**)
PSO with wind dispersion and confinement	[56]	94.41

Note: PSO result based on an error tolerance of ± 0.027 for the fitness function.

* Best results among different configurations.

** Results obtained on train/test sets of 1000/3177.

6. Discussion and conclusion

As a first balance sheet of this work, it can be concluded that investigating PSO-based classification is difficult. This stems from at least three reasons:

1. The wealth of material to investigate: It is difficult to stop the review process since new contributions are published every day, making it difficult to warrant the exhaustiveness of any survey.
2. The youth of PSC: No specific study on PSC has been published before.
3. The lack of taxonomic guidelines to organize the domain.

For all these reasons, the PSC survey presented in this work had to begin by proposing a taxonomic decomposition of PSC into plain and hybrid PSO-based categories, and then opt for a chronological order in order to describe the various works. An alternative approach to the survey could have been based on the various mechanisms used to create modified plain or hybrid POS-based classifiers. However, the great diversity of works in such case leads to too many sub-categories to cover.

Regarding the efficient use of PSC for high dimensional databases with large numbers of instances, De Falco et al. [6] pointed at the following potential limiting factors:

- Problem domains where the product number of instances by number of problem dimensions is important.
- Limitation of the PSO approach when the number of classes is important in multi class databases.

When considering the results achieved on the first three databases that were investigated in this work, it can be deducted that the apprehensions expressed in [6] do not always hold and that the enhanced PSO with dispersion and confinement mechanisms has good potential as a classification tool even for high

dimensioned problem spaces with a large number of instances and multiple classes. Moreover, the enhanced PSO performs better than the original works using neural networks, genetic classifiers and *k*-NN [58,59,62] for the three data sets.

Another important observation is that the interval confinement mechanism does not always guaranty good accuracy, since it is used to confine particle positions within the search space and not to diffuse them better. It performed very well for the first dataset but not so for the two others. On the other hand, wind dispersion with confinement performed well in all PSO experiments. Indeed, it permits a good exploration of the search space with no crossing of interval limits.

From the reported classification performances for various datasets, the results clearly indicate that PSO outperformed other classification methods in accuracy. Nevertheless, the time consumption of the training stage is substantially higher than with other techniques, or with continuous PSO-based classification as shown in the case of the Adult data set. The large value of learning time is to be expected when considering the Adult dataset, which is viewed as a large-scale and high-dimensional application in the literature [68]. Still, a speed up of the learning process is possible through the parallel implementation of the algorithm. Another possible enhancement of the approach is to consider it in the context of on-line, large-scale learning. Then, the PSO algorithm could be adapted in such a way that it runs through any large-scale dataset once.

We can also notice that despite the fact that [71] ranked CMC data set among the most difficult dataset to classify, the PSO approach performed successfully when applied to it. It the case of the Abalone dataset, PSO ranked second in classification performance and was a very close runner up to the GP-based classifiers as shown in Table 6.

Considering the achieved results on the six datasets, it can be concluded that PSC can be efficiently applied to classification problems with a large number of instances, both in continuous and mixed-attribute problem description spaces. Moreover, the obtained results are to the effect that PSC cannot only be applied successfully to more demanding problem domains, but also that it is a competitive alternative to well established classification techniques. In this respect, this work reinforces the usefulness of PSO for classification tasks. Furthermore, as an optimization tool for supervised classification, it requires no prior assumption about the distribution or dimensionality of the input data. This is an interesting advantage in comparison to classical methods such as linear discriminant functions.

It should be noted also that the typical PSO algorithm used is in global best style. It is characterized by a high interconnectivity between particles but a lower diversity comparatively to the local best style [83]. In the enhanced PSO algorithm that was introduced by [10] this situation is avoided by using the wind dispersion mechanism to ensure a nature-inspired diversification process.

Conflict of interest statement

There is no conflict of interest.

Acknowledgments

This work was possible thanks to the financial support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] R. Poli, Analysis of the publications on the applications of particle swarm optimisation, Journal of Artificial Evolution and Applications, 2008, http://dx.doi.org/10.1155/2008/685175, Article ID 685175.
[2] A. Abraham, He Guo, Hongbo Liu, Swarm intelligence: foundations, perspectives and applications in: Studies in Computational Intelligence (SCI), 26, Springer-Verlag, Berlin/Heidelberg, 2006 3–25.
[3] J. Kennedy, R. Eberhart (Eds.), Morgan Kaufmann, 2001.
[4] D. Merkle, M. Middendorf, Swarm intelligence, in: E.G. Burke, Kendall (Eds.), Introductory Tutorials in Optimisation, Search and Decision Support Methodology, Springer, 2005, pp. 401–435.
[5] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the fourth IEEE International Conference on Neural Networks, Perth, Australia, 1995, pp. 1942–1948.
[6] I. De Falco, A. Della Cioppa, E. Tarontino, Evaluation of Particle Swarm Optimization Effectiveness in Classification, Springer-Verlag, Berlin/Heidelberg, 2006, LNAI3849, pp. 164–171.
[7] Y. Owechko, S. Medasani, N. Srinivasa, Classifier swarms for human detection in infrared imagery, in: Proceedings of the 2004, IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW'04), 2004.
[8] M. O'Neill, A. Brabazon, Self-organizing swarm (SOSwarm): a particle swarm algorithm for unsupervised Learning, in: proceedings of IEEE Congress on Evolutionary Computation, July 16–21, Vancouver, BC, Canada, 2006, pp. 634–639.
[9] I. De Falco, A. Della Cioppa, E. Tarontino, Facing classification problems with particle swarm optimization, Applied Soft Computing 7 (2007) 652–658.
[10] N. Nouaouria, M. Boukadoum, Particle swarm classification for high dimensional data sets, in: Proceedings of 22th International IEEE Conference on Tools with Artificial Intelligence IEEE-ICTAI, 27–29 October 2010, vol. 1, Arras, France, 2010, pp. 87–93.
[11] G.S. Tewolde, D.M. Hanna, Particle swarm optimization for classification of breast cancer data using single and multisurface methods of data separation, in: Proceedings of IEEE EIT, 2007, pp. 443–446.
[12] M.G.H. Omran, A. Al-Sharhan, Barebones particle swarm methods for unsupervised image classification, in: Proceedings of IEEE Congress on Evolutionary Computation (CEC), September 2007, Singapore, 2007, pp. 3247–3252.
[13] A. Cervantes, I. Galvan, P. Isasi, An adaptive Michigan Approach PSO for nearest prototype classification, in: J. Mira, J.R. Alvarez (Eds.), Proceedings of IWINAC, 2007, Part II, LNCS 4528, Springer-Verlag, Berlin/Heidelberg, 2007, pp. 287–296.
[14] A. Cervantes, I. Galvan, P. Isasi, Michigan particle swarm optimization for prototype reduction in classification problems, New Generation Computing 27 (2009) 239–257.
[15] A. Cervantes, I. Galvan, P. Isasi, AMPSO: a new particle swarm method for nearest neighborhood classification, IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 39 (5) (2009).
[16] Chen Wei, Jun Sun, Yanrui Ding, Wei Fang, Wenbo Xu, Clustering of gene expression data with quantum-behaved particle swarm optimization, in: N.T. Nguyen, et al., (Eds.), IEA/AIE 2008, Springer-Verlag, Berlin/Heidelberg, 2008, pp. 388–396, LNAI 5027.
[17] Tsai Chi-Yang, Szu-Wei Yeh, A multiple objective particle swarm optimization approach for inventory classification, International Journal of Production Economics 114 (2008) 656–666.
[18] Lee Ee Ng, Mei Kuan Lim, Tomás Maul, Weng Kin Lai, Investigations into particle swarm optimization for multi-class shape recognition, in: Proceedings of ICONIP 2008, M. Köppen et al. (Eds.): Part II, LNCS 5507, Springer-Verlag, Berlin/Heidelberg, 2008, pp. 599–606, .
[19] Liu Ruochen, Xiaojuan Sun, LichengJiao, Particle swarm optimization based clustering: a comparison of different cluster validity indices, in: K. Li et al. (Eds.): Part II, CCIS 98, Proceedings of LSMS/ICSEE 2010, Berlin/Heidelberg, Springer-Verlag, 2010, pp. 66–72.
[20] R. Killani, K. S. Rao, S. Satapathy, G. PradhanK. R. Chandran, Effective document clustering with particle swarm optimization, in: B.K. Panigrahi et al. (Eds.): LNCS 6466, Proceedings of SEMCCO 2010, Springer-Verlag, Berlin/Heidelberg: 2010, pp. 623–629.
[21] S. Nebti, A. Boukerram, Handwritten digits recognition based on swarm optimization methods, in: F. Zavoral et al. (Eds.): Part I, CCIS 87, Proceedings of NDT 2010, Springer-Verlag, Berlin/Heidelberg, 2010, pp. 45–54.
[22] K. Chandramouli, E. Izquierdo, Image classification using chaotic particle swarm optimization, in: Proceedings of International Conference on Image Processing (ICIP '06).
[23] M.G.H. Omran, A.P. Engelbrecht, A. Salman, Dynamic clustering using particle swarm optimization with application in unsupervised image classification, Enformatika Transactions on Engineering, Computing and Technology1305-53139 (2005) 199–204.
[24] Gao1 Haichang, Boqin Feng, Yun Hou, Li Zhu , Training RBF neural network with hybrid particle swarm optimization, in: J. Wang et al. (Eds.): LNCS 3971, Proceedings of ISNN 2006 Springer-Verlag, Berlin/Heidelberg, 2006, pp. 577–583.
[25] H. Ichihashi, K. Honda, A. Notsu, K. Ohta, Fuzzy *c*-means classifier with particle swarm optimization, in:Proceedings of IEEE International Conference on Fuzzy Systems (FUZZ 2008), 2008, pp. 207–215.

[26] F. Melgani, Y. Bazi, Classification of electrocardiogram signals with support vector machines and particle swarm optimization, IEEE Transactions on Information Technology in Biomedicine 12 (5) (2008) 667–677.
[27] Hung Chih-Cheng, Hendri Purnawan, A hybrid rough *k*-means algorithm and particle swarm optimization for image classification, in: A. Gelbukh, E.F. Morales (Eds.), in: Proceedings of MICAI, Springer-Verlag, Berlin/Heidelberg, 2008, pp. 585–593, LNAI 5317.
[28] C.R. Hema, M.P. Paulraj, S. Yaacob, A.H. Adom, R. Nagarajan, Particle swarm optimization neural network based classification of mental tasks, in: N.A. Abu Osman, F. Ibrahim, W.A.B. Wan Abas, H.S. Abd Rahman, H.N. Ting (Eds.): 21, Biomed 2008 Proceedings, Springer-Verlag 2008, Berlin/Heidelberg, pp. 883–888.
[29] B. Biswal, P.K. Dash, B.K. Panigrahi, Power quality disturbance classification using fuzzy *c*-means algorithm and adaptive particle swarm optimization, IEEE Transactions on Industrial Electronics 56 (1) (2009).
[30] B. Dehuri, S. Mishra, S.B. Cho, in: M. Köppen, et al., (Eds.), A notable swarm approach to evolve neural networkfor classification in data mining, in: Proceedings of ICONIP, Springer-Verlag, Berlin/Heidelberg, 2008, pp. 1121–1128, Part I, LNCS 5506.
[31] J.E. Fieldsend, Optimizing decision trees using multi-objective particle swarm optimization, in: C.A. Coello Coello, et al., (Eds.), Swarm Intelligence for Multi-objective Prob., SCI 242, Springer-Verlag, Berlin/Heidelberg, 2009, pp. 93–114.
[32] H. Nuzly, N. Kasabov, Z. Michlovsky, S.M. Shamsuddin, String pattern recognition using evolving spiking neural networks and quantum inspired particle swarm optimization, in: C.S. Leung, M. Lee, J.H. Chan (Eds.), in: Proceedings of ICONIP, Springer-Verlag, Berlin/Heidelberg, 2009, pp. 611–619, Part II, LNCS 5864.
[33] A.B.S. Serapiao, J.R.P. Mendes, Classification of petroleum well drilling operations with a hybrid particle swarm/ant colony algorithm, in: B.C. Chien, et al., (Eds.), in: Proceedings of IEA/AIE, Springer-Verlag, Berlin/Heidelberg, 2009, pp. 301–310, LNAI 5579.
[34] Cui Yu, Fei Han, Shiguang Ju, Gene selection and PSO-BP classifier encoding a prior information, in: Y. Tan, Y. Shi, K.C. Tan (Eds.), in: Proceedings of ICSI, Springer-Verlag, Berlin/Heidelberg, 2010, pp. 335–342, Part II, LNCS 6146.
[35] J. Dheeb, T. Selvi, Bio inspired swarm algorithm for tumor detection in digital mammogram, in: B.K. Panigrahi, et al., (Eds.), in: Proceedings of SEMCCO, Springer-Verlag, Berlin/Heidelberg, 2010, pp. 404–415, LNCS 6466, pp..
[36] F. Al Obeidat, N. Belacel, J.A. Carretero, P. Mahanti, Automatic parameter settings for the proaftn classifier using hybrid particle swarm optimization, in: A. Farzindar, V. Keselj (Eds.), in: Proceedings of Canadian AI, Springer-Verlag, Berlin/Heidelberg, 2010, pp. 184–195, LNAI 6085, pp..
[37] R. Kohavi, J.H. George, The wrapper approach, in: Liu Huan, Motoda Hiroshi (Eds.), Feature Extraction, Construction and Selection: A Data Mining Perspective, Kluwer Academic Publishers, ISBN 0792381963, 1998.
[38] J. Fan, Y. Fan, High-dimensional classification using features annealed independence rules, Annals of Statistics 36 (6) (2008) 2605–2637.
[39] I.K. Fodor, A Survey of Dimension Reduction Techniques, Technical Report UCRL-ID-148494, Lawrence Livermore Nat'l Laboratory, Center for Applied Scientific Computing, 2002.
[40] H. Liu, A. Abraham, W. Zhang, A fuzzy adaptive turbulent particle swarm optimization, International Journal of Innovative Computing and Applications Archive 1 (1) (2007).
[41] L.S. Coelho, V. Mariani, A novel chaotic particle swarm optimization approach using Hénon map and implicit filtering local search for economic load dispatch, Chaos, Solitons and Fractals (39) (2009) 510–518.
[42] Jun-Jie Xue Wang, Sheng Ma, Dao-Wei Bi Wang, Distributed particle swarm optimization and simulated annealing for energy-efficient coverage in wireless sensor networks, Sensors1424-82207 (2007) 628–648.
[43] V.M. Saffarzadeh, P. Jafarzadeh, M. Mazloom, A hybrid approach using particle swarm optimization and simulated annealing for *n*-queen problem, Journal of World Academy of Science, Engineering and Technology 67 (2010) 974–978.
[44] M. Kessentini, H. Sahraoui, M. Boukadoum, O. Ben Omar, Search-based model transformation by example, Journal of Software and System Modeling Special issue on Models (2008) 1–18.
[45] M. Clerc, L'optimisation par Essaim Particulaire: Versions Paramétriques et Adaptatives, Hermes Science Publications, Lavoisier, Paris, 2005.
[46] N. Nouaouria, M. Boukadoum, A particle swarm optimization approach to mixed attribute data-set classification, in: Proceedings of IEEE Symposium on Swarm Intelligence, 978-1-61284-052-9, Paris, France, 2011, pp. 44–51.
[47] J. Kennedy, R. Eberhart, A discrete binary version of the particle swarm algorithm, in: IEEE Conference on Systems, Man, and Cybernetics, vol. 5 (1997), pp. 4104–4108.
[48] B. Al kazemi. C.K. Mohan, Multi-phase discrete particle swarm optimization, in: Fourth International Workshop on Frontiers in Evolutionary Algorithms, Kinsale, Ireland, 2002.
[49] S. Yang, M. Wang, L. Jiao, A quantum particle swarm optimization, in: Proceedings of CEC 2004, the Congress on Evolutionary Computing, vol. 1, 2004, pp. 320–324.
[50] G. Pampara, N. Franken. A.P. Engelbrecht, Combining particle swarm optimisation with angle modulation to solve binary problems, in: Proceedings of the IEEE Congress on Evolutionary Computing, vol. 1, 2005, pp. 89–96.
[51] H.A. Hassan, I.M. Yassin, A.K. Halim, A. Zabidi, Z.A. Majid, H.Z., Abidin, Logical effort using a novel discrete particle swarm optimization algorithm, in: Proceedings of Fifth International Colloquium on Signal Processing & its Applications (CSPA), 2009, 978-1-4244-4152-5/09.
[52] J. Pugh, A. Martinoli, Discrete multi-valued particle swarm optimization, in: Proceedings of IEEE Swarm Intelligence Symposium, vol. 1, 2006, pp. 103–110.
[53] J.A. Moreno-Pérez, J.P. Castro-Gutiérrez, F.J. Martinez-Garcia, B. Melian, J.M. Moreno-Vega, J. Ramos, Discrete particle swarm optimization for the p-median problem, in: Proceedings of the Seventh Metaheuristics International Conference, Montréal, Canada, 2007.
[54] S. Consoli, J.A. Moreno-Perez, K. Darby-Dowman, N. Mladenovic, Discrete particle swarm optimization for the minimum labelling Steiner tree problem, Natural Computing, http://dx.doi.org/10.1007/s11047-009-9137-9, 2009.
[55] N. Nouaouria, M. Boukadoum, A particle swarm optimization approach for case retrieval stage, in: M. Bramer, M. Petridis, A. Hopgood (Eds.), Research and Development in Intelligent Systems XXVII Proceedings of AI-2010, The Thirtieth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, 14–16 December 2010, Cambridge United Kingdom, 2010, pp. 209–222.
[56] N. Nouaouria, M. Boukadoum, Improved global-best particle swarm optimization algorithm with mixed-attribute data classification capability, Applied Soft Computing, Manuscript ID: ASOC-D, 2012 12-00385.
[57] A. Asuncion, D.J. Newman, UCI Machine Learning Repository 〈http://www.ics.uci.edu/∼mlearn/MLRepository.html〉. Irvine, CA: University of California, School of Information and Computer Science, 2007.
[58] H. Naoum, M. Boukadoum, C. Joseph, D. Starikov, A. Bensaoula, Intelligent classifier module for fluorescence based measurements, in: Proceedings of International Workshop on Signal Processing and its Applications (WoSPA2008), 18–20 March 2008, Sharjah, UAE, 2008.
[59] P.W. Frey, D.J. Slate, Letter recognition using Holland-style adaptive classifiers, Machine Learning 6 (1991) #2.
[60] A.M. Bagirov, J. Ugon, D. Webb, An incremental approach for the construction of a piecewise linear classifier, in: Proceedings of The XIII International Conference "Applied Stochastic Models and Data Analysis" (ASMDA-2009), June 30–July 3, 2009, Vilnius, LITHUANIA, 2009, pp. 507–511.
[61] A. Astorino, M. Gaudioso, Polyhedral separability through successive LP, Journal of Optimization Theory and Applications 112 (2) (2000) 265–293.
[62] E. Alpaydin, C. Kaynak, Cascading classifiers, Kybernetika 34 (4) (1998) 369–374.
[63] E. Cant-Paz, C. Kamath, Using Evolutionary Algorithms to Induce Oblique Decision Trees, Technical Report UCRL-JC-137202, Center for Applied Scientific Computing Lawrence Livermore National Laboratory, 2000.
[64] A. Demiriz, K.P. Bennett, J. Shawe-Taylor, Linear programming boosting via column generation, Machine Learning, 46, 2002 225–254.
[65] R. Kohavi, Scaling up the accuracy of Naive–Bayes classifiers: a decision-tree Hybrid, in: E. Simoudis, J. Han, U. Fayyad (Eds.): Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD, AAAI Press, 1996, pp. 202–207.
[66] R. Kohavi, B. Becker, D. Sommerfield, Improving simple Bayes, in: M. van Someren, G. Widmer, (Eds.), Proceedings of the Nineth European Conference on Machine Learning, Springer-Verlag, Heidelberg, 1997, pp. 78–87.
[67] J. Cheng, R. Greiner, Learning Bayesian belief network classifiers: algorithms and system, in: Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence, 2001, pp. 141–151.
[68] L. Hoegaerts, J. Suykens, J. Vandewalle, B. De Moor, Primal space sparse kernel partial least squares regression for large scale problems, in: IEEE Proceedings of the International Joint Conference on Neural Networks (IJCNN'2004), pp. 561–566.
[69] Kai-Min Chung Kao Wei-Chun, L. Assun, Chih-Jen Lin, Decomposition methods for linear support vector machines, Neural Computation 16 (8) (2004) 1689–1704.
[70] H. Abbasian, C. Drummond, N. Japkowicz, S. Matwin, Robustness of classifiers to changing environments, Advances in Artificial Intelligence Lecture Notes in Computer Science 6085/2010 (2010) 232–243, http://dx.doi.org/10.1007/978-3-642-13059-5_23.
[71] T.-S. Lim, W.-Y. Loh, Y.-S. Shih, A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms, Machine Learning 40 (2000) 203–228.
[72] Bose Kooperberg, Stone, polychotomous regression, Journal of American Statistical Association 92 (1997) 117–127.
[73] W.Y. Loh, Y.S. Shih, Split selection methods for classification trees, Statistica Sinica 7 (1997) 815–840.
[74] K.J. Cios, L.A. Kurgan, CLIP4: Hybrid inductive machine learning algorithm that generates inequality rules, Information Sciences Journal 163 (2004) (2004) 37–83.
[75] M. Rochas, P. Cortez, J. Nevez, Evolutionary neural network learning algorithms for changing environments, WSEAS Transactions on Systems1109-27773 (2) (2004) 596–601.
[76] J. Bacardit, M.V. Butz, Data Mining in Learning Classifier Systems: Comparing XCS with GAssist, in: Learning Classifier Systems, Lecture Notes in Computer Science, 2007, 4399/2007, 282–290, 10.1007/978-3-540-71231-2_19.
[77] S. Ray, D. Page, Generalized skewing for functions with continuous and nominal attributes, in: Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 2005.
[78] L. Uysal, H.Altay Gu, Instance-based regression by partitioning feature projections, Applied Intelligence Journal, 21, 57–79.
[79] L. Uysal, H.A. Guvenir, An overview of regression techniques for knowledge discovery, Knowledge Engineering Review 14 (1999) 1–22.
[80] K.M. Ting, I.H. Witten, Issues in stacked generalization, Journal of Artificial Intelligence Research 10 (1999) 271–289.

[81] E. Snelson, C.E. Rasmussen, Z. Ghahramani, Warped Gaussian Processes, in: Advances in Neural Information Processing Systems 16 (NIPS-2003). MIT Press, Cambridge, MA, 2003.
[82] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, MIT Press, 2006 ISBN-10 0-262-18253-X, ISBN-13 978-0-262-18253-9.
[83] A.P. Engelbrecht, Computational Intelligence: An Introduction, John Willey & Sons Editions, 2007.
[84] R. Horst, H. Tuy, Global optimization: deterministic approaches, third ed., Springer, Berlin, 2003, Revised and Enlarged Edition.

Nabila Nouaouria obtained her doctorate of state (2006) in artificial intelligence at university of Annaba, Algeria, where she worked as associate professor from 1996 to 2006. She is currently a Ph.D. candidate at University of Quebec at Montreal (UQAM), Canada, in the cognitive informatics program.

Mounir Boukadoum is Professor of Microelectronics Engineering at the Department of Computer Science, University of Quebec at Montreal (UQAM), Canada. He studied physics at the University of Algiers (Algeria) from 1973 to 1976. He then switched to electrical engineering and received the Master's degree from The Stevens Institute of Technology in 1978 and the Ph.D. degree from The University of Houston in 1983. He joined UQAM in 1984. He was Director of Microelectronics Programs at UQAM for three three–year mandates in 1994, 1998 and 2001, Director of the Ph.D. Program in Cognitive Informatics from 2006 to 2009, and Associate Department Chair for Research in 2007–2008. He currently chairs the MicroelectronicsPrototyping Research Laboratory at UQAM, is Executive Member of the Microsystems Strategic Alliance of Quebec (ReSMiQ), and President of the Montreal chapter of the IEEE Computational Intelligence Society.

Robert Proulx obtained a doctorate in psychology (1986) University of Montreal, a Master degree (1980) and a baccalaureat with Honours in psychology (1977) at UQAM. Pr. Proulx is professor in the Psychology Department since 1978 and Dean of the Faculty of Humanities since 1999. In addition to leading the Department of Psychology from 1994 to 1997, Pr. Proulx was acting director of two other departments, Religious Studies and Linguistics and Language Teaching. Pr. Proulx worked also as a professor and researcher in artificial intelligence and cognitive science. He is member of many scientific committees and boards. He is currently Vice-Rector of Academic Life at UQAM.