Applied Soft Computing 21 (2014) 554–567

Contents lists available at ScienceDirect

Applied Soft Computing

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

Improved global-best particle swarm optimization algorithm with mixed-attribute data classification capability

Nabila Nouaouria*, Mounir Boukadoum

Computer Science Department, UQAM, CP. 8888, Succ. C.-V., Montreal, QC H3P3P8, Canada

ARTICLE INFO

Article history:
Received 11 March 2012
Received in revised form 4 March 2014
Accepted 9 April 2014
Available online 24 April 2014

Keywords:
Mixed attribute data sets
Particle swarm optimization
Supervised learning
Classification

ABSTRACT

This paper describes a novel Particle Swarm Optimization (PSO)-based classification algorithm with improved capabilities in comparison to several alternatives. The algorithm uses a new particle-position update mechanism and a new way to handle mixed-attribute data based on particle position interpretation. The new position update mechanism combines particle confinement and dispersion for improved search space coverage, and the proposed interpretation mechanism uses the frequencies of non numerical attributes instead of integer mappings. As our experimental results have shown, this leads to better cost function evaluation in the description space and subsequently enhanced processing of mixed-attribute data by the PSO algorithm. Our experimental setup consisted of three large benchmark databases, and the obtained recognition accuracies were better than those obtained with well-known classifiers.

1. Introduction

Heterogeneous input data consisting of a mixture of discrete, continuous and nominal variables are frequent in classification tasks. For instance, when sorting patients into diagnostic and prognostic groups (healthy and sick) in a hospital, to decide whether to admit them in or treat them as outpatients, the outcome typically relies on indicators that include continuous (e.g. temperature), nominal (e.g. presence or absence of some symptoms) 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). Finally, one can also think of the technical specifications of an electronic circuit where mixed-attribute data are common.

When dealing with mixed data, classifiers typically conduct a preliminary coding stage to map the non numerical values into integer enumerations. Two challenges need addressing when doing so: How to establish an order relation on the transformed data and the potential knowledge representation bias caused by a flat (unweighted) representation of the non-numeric data, with the result that the relative importance of the individual non-numeric values is lost during the coding process. In this paper, we present an approach that avoids these problems by interpreting instead of straight coding. The interpretation mechanism inherently reproduces the weighting semantics of the non-numeric data, and it is easily integrated in a metaheuristic-based classification approach, particle swarm optimization (PSO) in this work.

In the remainder of this paper, Section 2 is a brief literature review of PSO approaches for classification. Section 3 provides a concise survey of how previous PSO approaches have dealt with continuous and discrete data. Section 4 presents PSO adaptation for classification tasks and our approach with its computational cost analysis, followed by the experimental results achieved on three large benchmark data sets in Section 5. Related works are discussed in Section 6, and a conclusion ends the paper.

2. A brief survey of PSO in the classification task

Efforts that seek to apply PSO to more diversified problem areas are increasing. Poli [1] defines 26 different categories of applications where PSO has been used successfully. The categorization was the result of analyzing more than eleven hundred publications on PSO stored in the IEEE Xplore database and the study revealed that clustering, classification and data mining represented 4.3% of the total production.

PSO has already proven its effectiveness for classification tasks (see De Falco et al. [2] or Section 6). The literature indicates that a particle swarm optimizer constitutes a suitable and competitive technique for such an endeavor, and that it can be successfully

* Corresponding author. Tel.: +1 5143445098.
E-mail addresses: nouaouria.nabila@gmail.com (N. Nouaouria), boukadoum.mounir@uqam.ca (M. Boukadoum).

http://dx.doi.org/10.1016/j.asoc.2014.04.018
1568-4946/

applied to demanding problem domains, particularly when accurate yet comprehensible classifiers, fit for dynamic, distributed environments are required [3]. Below is a chronological survey of some recent studies that used PSO for classification tasks[1]:

In [4], PSO is extended by using 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 [5], the authors describe a self-organizing particle swarm algorithm, SOSwarm that adopts unsupervised learning. The input vectors are projected onto a lower dimensional map space, hence 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, therefore, 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 [34] (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.

De Falco et al. [2] offer an evaluation of PSO's efficiency for a set of classification tasks. PSO was applied to data from nine different databases taken from the UCI repository, and the obtained results were compared to those provided by nine classical classification algorithms available within the Waikato Environment for Knowledge Analysis (WEKA) system, release 3.4 [60]. The alternative classification methods were as follows: 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 in either technique were those set by default in WEKA. The obtained results were that PSO had the best accuracy for three out of nine challenged problems. Some relationships between problem size and PSO performance were also hypothesized from the experimental results [2,6] to the 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 [7] where remedial mechanisms for high dimensional datasets were proposed.

In [8–10], 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 data sets. 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 where a 1-NN classifier did not perform well. In particular, AMPSO significantly outperformed the other algorithms on the Glass Identification data set, with more than 10% improvement in accuracy on average.

The authors in [11] applied PSO to inventory classification and developed a flexible classification algorithm that can be utilized as a single objective algorithm for cost minimization, demand correlation maximization, or inventory turnover ratio maximization. It can also operate as a multi-objective algorithm that takes into account multiple measures at the same time. The algorithm can determine the best number of classification groups. Numerical studies were conducted, and the classification performance of the PSO algorithm was comparable to other approaches.

In [12], 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.

3. Discrete PSO and limitations

This section gives a recap of continuous PSO in Section 3.1 before focusing on discrete PSO in Section 3.2.

3.1. PSO Background

The roots of PSO lie in ethological metaphors for computing models [14–16]. For example, the coordinated search that lets a flock of birds spot a promising food location can be modeled with simple rules for information sharing between individuals. Such behavior inspired Kennedy and Eberhart (see [14,17]) to develop PSO as a method for function optimization. In essence, the PSO algorithm maintains a population of particles (the 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 through the search space looking for the minimum (or maximum) of a given objective function. In the bird analogy, this function would be a measure of the quality or quantity of food at each place, and the particle swarm would search for the place with the best and/or most abundant food supply. The movements of a particle depend only on its velocity and the memory of locations where good solutions have already been found by the particle itself or by other (neighboring) particles in the swarm. This is again in analogy to bird flocking where every individual makes its decisions based on cognitive aspects (good solutions found by the particle itself) and social aspects (good solutions found by other particles). It should be noted that, unlike many deterministic methods for continuous function optimization, PSO uses no gradient information to find solutions.[2]

When the neighborhood of a particle 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 usually referred to as *lbest* PSO. The performance of each particle (i.e. how close the particle is to the target) is measured by a cost function whose form depends on the optimization problem.

More formally, a PSO algorithm is based on a swarm of M individuals or particles, each representing a potential solution to a problem with N dimensions. Its genotype consists of $2N$ parameters, the first half representing the coordinates of a particle in the N-dimensional problem space and the second half the corresponding velocity components. A particle moves with an adaptable

[1] Only studies using plain PSO are mentioned; PSO is commonly combined with other classification approaches in the literature, mainly neural networks.

[2] As a result, there is no continuous error function requirement for computing a derivative.