Applied Soft Computing 9 (2009) 470–476

Contents lists available at ScienceDirect

Applied Soft Computing

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

Particle swarm optimized multiple regression linear model for data classification

Suresh Chandra Satapathy [a,*], J.V.R. Murthy [b], P.V.G.D. Prasad Reddy [c], B.B. Misra [d], P.K. Dash [d], G. Panda [e]

[a] Anil Neerukonda Institute of Technology and Sciences, Vishakapatnam, AP, India
[b] JNTU College of Engineering, Kakinada, India
[c] College of Engineering, AU, India
[d] College of Engineering, Bhubaneswar, India
[e] National Institute of Technology, Rourkela, India

ARTICLE INFO

Article history:
Received 1 February 2006
Accepted 29 May 2008
Available online 22 June 2008

Keywords:
Particle swarm optimization (PSO)
Least square estimation
Multiple linear regression

ABSTRACT

This paper presents a new data classification method based on particle swarm optimization (PSO) techniques. The paper discusses the building of a classifier model based on multiple regression linear approach. The coefficients of multiple regression linear models (MRLMs) are estimated using least square estimation technique and PSO techniques for percentage of correct classification performance comparisons. The mathematical models are developed for many real world datasets collected from UCI machine repository. The mathematical models give the user an insight into how the attributes are interrelated to predict the class membership. The proposed approach is illustrated on many real data sets for classification purposes. The comparison results on the illustrative examples show that the PSO based approach is superior to traditional least square approach in classifying multi-class data sets.

1. Introduction

Data classification plays a major role in any pattern recognition problem. It is a supervised learning strategy which emphasizes on building models able to assign new instances to one of a set of well-defined classes. There has been wide range of machine learning and statistical methods for solving classification problems. Many algorithms have been developed including classical methods such as linear discriminant analysis and Bayesian classifiers, statistical techniques such as MARS (multivariate adaptive regression splines), machine learning approaches for decision trees, etc. including C4.5, CART, C5, bayes trees and neural network approaches such as multiplier perceptron and neural trees [1–8]. Approaches like fuzzy logic, support vector machine (SVM), tolerant rough sets, principal component analysis (PCA), linear programming also have been very popular for data classification problems [9].

Some of the classification techniques mentioned above work well when the classes are linearly separable. However, in many real world problems the data may not be linearly separable and also data are very closely spaced and therefore a highly nonlinear decision boundary is required to separate the data. Techniques like neural network (NN), SVM, Fuzzy logic are very useful approaches for such cases. However, in many cases it is desired to find a simple classifier which gives the user a rough, but understandable insight into how the data attributes relates to class memberships. This objective can be achieved if it is possible to learn relationship hidden in data and express them in mathematical manner. There have been some attempts to solve classification problems using mathematical programming of linear discriminant analysis [10]. Very recently genetic programming (GP) [11] has been used for developing a mathematical model automatically for classifying multi-class problems [12–15]. GP is an effective approach in discovering the underlying relationship among data and express the relationship among attributes in an understandable manner for classification problem. But the resultant mathematical models obtained using [13,15] require many arithmetic operations while predicting the class for data sets having many features and many classes.

The objective of this paper is to present an effective mathematical model based on linear regression for multi-class data classification problem. We discuss our approach in developing multiple regression linear models (MRLMs) for different real data sets. The coefficients associated with the MRLM are estimated separately by least square estimation (LSE) [16,17] method and the classification accuracies are determined for all datasets separately. An evolutionary approach called particle swarm optimization (PSO) [19] is then used to estimate the coefficients of MRLM for each dataset and the classification accuracies are computed. The comparisons are made for above two approaches. It is shown that PSO approach outperforms the LSE approach in terms of giving better classification accuracy. Finally mathematical models are presented as illustrations for few datasets to show the inter-relationship existing among attributes of respective datasets.

* Corresponding author. Tel.: +91 8933225083x135; fax: +91 8933226395.
E-mail address: sureshsatapathy@ieee.org (S.C. Satapathy).

1568-4946/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2008.05.007

The rest of the paper is organized as follows. In Section 2 the MRLM is briefly discussed with LSE technique. Section 3 describes the basics of PSO. Data set description and simulation results are given in Section 4. Section 5 gives conclusion and some direction for future research.

2. Multiple regression linear model and least square estimator

Multiple regressions are an extension of linear regression involving more that one predictable variable. The model based on multiple regressions is here known as multiple regression linear model (MRLM). MRLM attempts to model the relationship between two or more explanatory variables and a response variable by fitting linear equation to observed data. Every value of the independent variable x is associated with a value of the dependent variable y. The population regression line for p explanatory variables $x_1,x_2,\ldots,x_p$ is defined to be

$$Y = c_0 + c_1x_1 + c_2x_2 + \cdots + c_px_p \quad (1)$$

In our study $x_1,x_2,\ldots,x_p$ represent the parameters or attributes of the data sets under investigation and the response Y is the class value for a particular instance in the data set.

The method of LSE can be applied here to solve for $c_0,c_1,c_2,\ldots,c_p$. In this method the best-fitting line for the observed data is calculated by minimizing the sum of squares of the vertical variations from each data point to the line. The LSE technique is explained in [16,17]. The brief description is given below.

Let the input and output data for training be represented in the following manner

$$\begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1m} & y_1 \\ x_{21} & x_{22} & \ldots & x_{2m} & y_2 \\ . & . & . & . & . \\ x_{n1} & x_{n2} & \ldots & x_{nm} & y_n \end{bmatrix}$$

In general, it is expressed as

$$(X_i, y_i) = (x_{1i}, x_{2i}, \ldots, x_{mi}, y_i)$$

where n: number of records; m: number of features.

The input and output relationship of the above data can be expressed in multiple regression linear model in the following manner as per (1).

$$\begin{bmatrix} y_1 = c_0 + c_1x_{11} + c_2x_{12} + \cdots + c_mx_{1m} \\ y_2 = c_0 + c_1x_{21} + c_2x_{22} + \cdots + c_mx_{2m} \\ \cdots \\ y_n = c_0 + c_1x_{n1} + c_2x_{n2} + \cdots + c_mx_{nm} \end{bmatrix} \quad (2)$$

The principle of least square minimizes the residual error between the estimated value and the desired value by choosing suitable values for co-efficient such as $c_0,c_1,c_2\ldots c_m$. The residual error can be expressed as follows

$$d_i = y_i - (c_0 + c_1x_{i1} + c_2x_{i2} + \cdots + c_mx_{im}) \quad (3)$$

The equations for the least square are

$$\prod = d_1^2 + d_2^2 + \cdots + d_n^2 = \sum_{i=1}^{n} d_i^2$$

$$= \sum_{i=1}^{n} [y_i - (c_0 + c_1x_{i1} + c_2x_{i2} + \cdots + c_mx_{im})]^2 \quad (4)$$

3. Particle swarm optimization

The particle swarm algorithm [19] is an optimization technique inspired by the metaphor of social interaction observed among insects or animals. The kind of social interaction modelled within a PSO is used to guide a population of individuals (so called particles) moving towards the most promising area of the search space. In a PSO algorithm, each particle is a candidate solution equivalent to a point in a d-dimensional space, so the i-th particle can be represented as $X_i = (x_{i1},x_{i2},\ldots,x_{id})$. Each particle "flies" through the search space, depending on two important factors, $P_i = (p_{i1},p_{i2},\ldots,p_{id})$, the best position the current particle has found so far refereed as *pbest*; and $P_g = (p_{g1},p_{g2},\ldots,p_{gd})$, the global best position identified from the entire population (or within a neighbourhood) refereed as *gbest*.

The rate of position change of the i-th particle is given by its velocity $V_i = (v_{i1},v_{i2},\ldots,v_{id})$. Eq. (5) updates the velocity for each particle in the next iteration step, whereas Eq. (6) updates each particle's position in the search space [11]:

$$v_{id}(t) = wv_{id}(t-1) + c_1(p_{id} - x_{id}(t-1)) + c_2(p_{gd} - x_{id}(t-1)) \quad (5)$$

$$x_{id}(t) = x_{id}(t-1) + v_{id}(t) \quad (6)$$

where w is inertia weight, c_1 and c_2 are acceleration coefficients.

Two common approaches of choosing p_g are known as *gbest* and *lbest* methods. In the *gbest* approach, the position of each particle in the search space is influenced by the best-fit particle in the entire population; whereas the *lbest* approach only allows each particle to be influenced by a fitter particle chosen from its neighbourhood. Kennedy and Mendes et al studied PSOs with various population topologies [20], and have shown that certain population structures could give superior performance over certain optimization functions.

There is no hard and fast rule as to how many particles should be used to solve a specific problem. A large number of particles allow the algorithm to explore the search space faster; however, the fitness function needs to be evaluated for each particle, so the number of particles will have a huge impact on the speed at which the simulation will run. Generally speaking, as the complexity of the search space increases, so should the number of particles.

The inertia weight, w, in the velocity vector update Eq. (5), is a scaling variable that controls the influence of the previous velocity when calculating the new velocity. Inertia weight values larger than one will typically cause the particle to accelerate and explore larger regions of the search space, while smaller values will cause the particle to gradually slow down and do a finer search of a region [18]. Many algorithms tend to decrease the inertia weight over time, allowing particles to initially roam a larger area in search of optima, and then to gradually do finer searches [18].

An early addition to the basic PSO algorithm was to place an upper limit on the velocity of a particle to prevent particles from moving too rapidly through search space. Clerc and Kennedy later proved that multiplying the velocity vector with a so-called constriction coefficient made velocity clamping unnecessary [19]. The constriction coefficient is a factor of the local and global component variables, for which the sum of the two has to be larger than four for the rule to apply.

The pseudo code of the PSO procedure is as follows

```
For each particle
 Initialize particle
End
Do
  For each particle
    Calculate fitness value
    If the fitness value is better than the best fitness value (pbest) in history
      Set current value as the new pbest
  End
  Choose the particle with the best fitness value of all the particles as the gbest
  For each particle
    Calculate particle velocity according equation (5)
    Update particle position according equation (6)
  End
While maximum iterations or minimum error criteria is not attained
```