Applied Soft Computing 11 (2011) 3106–3113

Contents lists available at ScienceDirect

Applied Soft Computing

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

A condensed polynomial neural network for classification using swarm intelligence

S. Dehuri[a,*], B.B. Misra[b], A. Ghosh[c], S.-B. Cho[d]

[a] Department of Information and Communication Technology, Fakir Mohan University, Vyasa Vihar, Balasore 756019, Orissa, India
[b] Department of Computer Science and Engineering, College of Engineering, Bhubaneswar 751024, Orissa, India
[c] Machine Intelligence Unit and Center for Soft Computing Research, Indian Statistical Institute Kolkata, 203 B.T. Road, Kolkata 700108, India
[d] Soft Computing Laboratory, Department of Computer Science, Yonsei University, 134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, Republic of Korea

ARTICLE INFO

Article history:
Received 29 April 2008
Received in revised form 11 August 2010
Accepted 12 December 2010
Available online 17 December 2010

Keywords:
Polynomial neural network
Classification
Particle swarm optimization
Partial descriptor
Data mining

ABSTRACT

A novel condensed polynomial neural network using particle swarm optimization (PSO) technique is proposed for the task of classification in this paper. In solving classification task classical algorithms such as polynomial neural network (PNN) and its variants need more computational time as the partial descriptions (PDs) grow over the training period layer-by-layer and make the network very complex. Unlike PNN the proposed network needs to generate the partial description for a single layer. The discrete PSO (DPSO) is used to select a relevant set of PDs as well as features with a hope to get better accuracy, which are in turn fed to the output neuron. The weights associated with the links from hidden to output neuron is optimized by PSO for continuous domain (CPSO). Performance of this model is compared with the results obtained from PNN. Simulation result shows that the performance of this model both in processing time and accuracy, is encouraging for harnessing its power in domain with large and complex data particularly in data mining area.

1. Introduction

In conjunction with the exponential growth of the information and communication technologies, we have witnessed a proliferation of databases with varying sizes of different degrees of complexities. Nevertheless, as the number of instances and its dimension grows, it is not that easy to analyze and retrieve high-level knowledge from the same databases. There are not as many off-the-shelf solutions for data analysis as there are for database creation and management; furthermore, they are pretty harder to suit to our needs. Data mining comprehend the actions of (semi) automatically looking for, identifying, validating, and using for decision making in data that might be categorized into classification, clustering, and association rule mining. In this paper we are giving emphasis on the problem of classification. Classification [1–4] is also treated as a challenging problem in pattern recognition and forecasting.

The goal of classification is to assign a class label from a predefined set of classes to an unknown sample based on the model of preference. Therefore, classification is based on some discovered model, which forms an important piece of knowledge about the application domain. Neural network based classifiers like multilayer perceptron with back propagation learning are available for classification. The reasons why it is being not yet exploited in data mining area are due to its longer training time, difficult to decide the number of neurons and number of layers and finally its black box nature, where knowledge learned is concealed with large number of connections.

To alleviate the shortcomings of NNs, polynomial neural network (PNN) based on Group Method of Data Handling (GMDH) approach suggested by Ivakhnenko [5–7] can be used for classification purposes. Polynomial neural networks are multi-layer partial descriptions (PDs) which produce high order multivariate polynomial mappings. The approach is based on evolutionary strategy where PNN generates populations or layers of PDs and then trains and selects those PDs, which provide the best classification. During learning the PNN model grows the new population of PDs and the number of layers until a predefined criterion is met. Thereby the complexity of the architecture increases and it is very difficult to comprehend by human user. Such models can be comprehensively described by a set of short-term polynomials. Coefficients of PNN can be estimated by least square fit.

The network architecture grows depending on the number of input features, PNN model selected, number of layers required, and the number of PD's preserved in each layer. In turn the architecture becomes very complex, requires huge memory and computation time. Hence to cope with this problem of PNN we propose a con-

* Corresponding author. Tel.: +91 94372 89873; fax: +82 2 365 2579.
E-mail addresses: satchi.lapa@gmail.com (S. Dehuri), misra_bijan@yahoo.co.in (B.B. Misra), ash@isical.ac.in (A. Ghosh), sbcho@yonsei.ac.kr (S.-B. Cho).

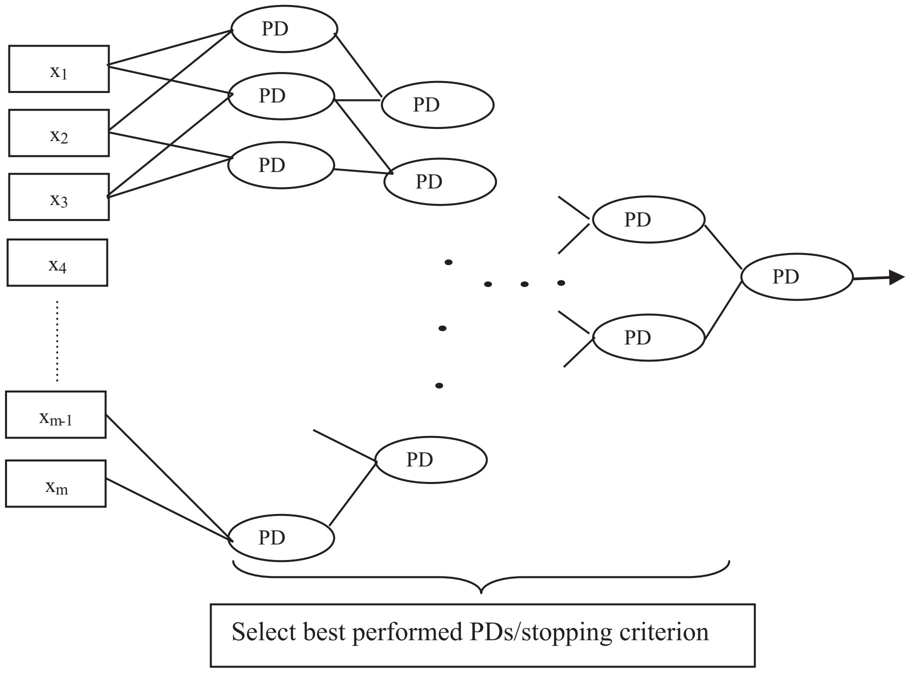


Fig. 1. Basic architecture of PNN model.

densed PNN model, which is a three layer architecture: input layer contains only the input features, hidden layer contains PDs and output layer contains only one neuron. We select an optimal set of PD's generated in the hidden layer along with the input features using the discrete PSO (DPSO) technique [8–10]. This optimal set is fed to the output layer. In conjunction the weights between hidden layer and output layer are optimized by PSO for continuous domain (CPSO). Why both version of PSO is chosen? The reason is that compared to other evolutionary algorithms (e.g. GAs), DPSO and CPSO is easy to implement and require very few parameters to adjust [11].

The rest of the paper is organized as follows. Section 2 describes the basics of PNN. In Section 3, PSO is discussed. The proposed model is formulated and discussed in Section 4. In Section 5, simulation result of the model is presented. Section 6 summarizes this paper with a prospect of future research directions.

2. Polynomial neural network

2.1. PNN architecture

The PNN architecture is based on the Group Method of Data Handling (GMDH) [12]. GMDH was developed by Ivakhnenko in late 1960s to identify non-linear relationship between input and output variables. However, there are several drawbacks associated with the GMDH such as its limited generic structure and overly complex network, and hence prompted a new class of polynomial neural networks (PNNs). In summary, these networks come with a high level of flexibility as each PD can have a different number of input variables as well as exploit a different order of polynomial (say linear, quadratic, cubic, etc.). Unlike neural networks whose topologies are commonly fixed prior to all detailed (parametric) learning, the PNN architecture is not fixed in advance but becomes fully optimized (both structurally and parametrically).

There are various types of PNN topology are developed so far, however it is worth noting to cover the basic one for getting more concrete idea in later part of the paper. The PNN architecture utilizes a class of polynomials such as linear, quadratic and cubic. By choosing the most significant number of variables and an order of the polynomial among these available forms, we can obtain the best ones from the extracted PDs according to selected nodes of each layer. Additional layers are generated until the best performance of the extended model has been reached. Such methodology leads to an optimal PNN structure. Let us assume that the input–output of the data is given in the following form:

$(X_i, y_i) = (x_{i1}, x_{i2}, ..., x_{im}, y_i)$, where $i = 1, 2, 3, \ldots, n$, n is the number of samples and m is the number of features. In matrix form we represent as follows:

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

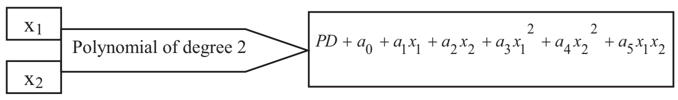
The input–output relationship of the above data by PNN model can be described in the following manner: $y = f(x_1, x_2, \ldots, x_m)$.

The estimated output of variables can be approximated by Volterra functional series, the discrete form of which is Kolmogorov–Gabor polynomial [4] and is written as follows.

$$y = a_0 + \sum_{1 \le i \le m} a_i x_i + \sum_{1 \le i,j \le m} a_{ij} x_i x_j + \sum_{1 \le i,j,k \le m} a_{ijk} x_i x_j x_k + \cdots \tag{1}$$

where a_k denotes the coefficients or weights of the Kolmogorov–Gabor polynomial and **x** vector is the input variables. The architecture of the basic PNN is shown in Fig. 1.

Each PDs the basic building block of the PNN model is shown in Fig. 2.



x_1
x_2
Polynomial of degree 2
$PD + a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1^2 + a_4 x_2^2 + a_5 x_1 x_2$

Fig. 2. Basic building blocks of PNN architecture.