ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 11, No. 2, 2016, *pp*. 129-135

General Regression Neural Network Optimization for Handwritten Persian Digits recognition Using Particle Swarm Optimization

Mohammad Masoud Javidi [1], Rahim Gholami Shooli [2]
[1] *Department of Computer, Shahid Bahonar University,Kerman, Iran, E-mail:javidi@uk.ac.ir, Tel.: +98-913-140 6784.*
[2] *Department of Computer, Shahid Bahonar University,Kerman, Iran, E-mail:gholami.r@math.uk.ac.ir*

(*Received October 28, 2015, accepted February 27, 2016)*

Abstract. In this paper an optimization algorithm based on Particle Swarm Optimization algorithm is used for handwritten Persian digits recognition with General Regression Neural Network .The system uses image zoning for the digit recognition. General Regression Neural Network accuracy depends on the centers and widths of the hidden layer neuron basis functions (neuron spread). In this paper we use Particle Swarm Optimization algorithm to determine General Regression Neural Network hidden layer spread. Results show that the optimized General Regression Neural Network provides higher recognition ability compared with that of unoptimized General Regression Neural Network.

Keywords: Particle Swarm Optimization, General regression neural networks, Pattern recognition, Farsi digits.

1. Introduction

Particle swarm optimizers (PSO) are optimization algorithms, modeled after the social behavior of flocks of birds [1]. PSO is a population based search process where individuals, referred to as particles, are grouped into a swarm. Each particle in the swarm represents a candidate solution to the optimization problem.

The General Regression Neural Network (GRNN) which is a kind of radial basis function (RBF) networks was developed by Specht (1991) [2] and is a powerful regression tool with a dynamic network structure. The GRNN was applied to solve a variety of problems like prediction, image processing, control, plant process modelling or general mapping problems. Recently, general regression neural network (GRNN) is successfully used in pattern recognition and image processing due to the advantages on fast learning and convergence to the optimal regression surface as the number of samples becomes very large [9].In Chen, Leung (2004) [12], GRNN is used to error correction in foreign exchange forecasting. In Kim et al. (2010), genetic algorithm is used to optimize GRNN to design optical lens controller. In Polat and Yildirim (2007) [9], genetic algorithm used to optimize GRNN to pattern recognition.

In this study we want to optimize GRNN using Particle swarm optimization (PSO) for Persian digit recognition. Here, Particle swarm optimizer is used to determine the spread value in neural network. Next Section 2 gives an overview of GRNN structure. In Section 3, a brief summary of Particle swarm optimization (PSO) is presented. In Section 4, simulation about how spread is selected is mentioned. And in section 5, results are given.

2. General Regression Neural Network (GRNN)

General Regression Neural Network (GRNN) was developed by Specht (1991) [2] does not require an iterative training procedure as in back propagation method. It approximates any arbitrary function between input and output vectors, drawing the function estimate directly from the training data. The GRNN is used for estimation of continuous variables, as in standard regression techniques. By definition, the regression of a dependent variable *y* on an independent *x* estimates the most probable value for *y*, given *x* and a training set. The regression method will produce the estimated value of *y* which minimizes the mean-squared error.

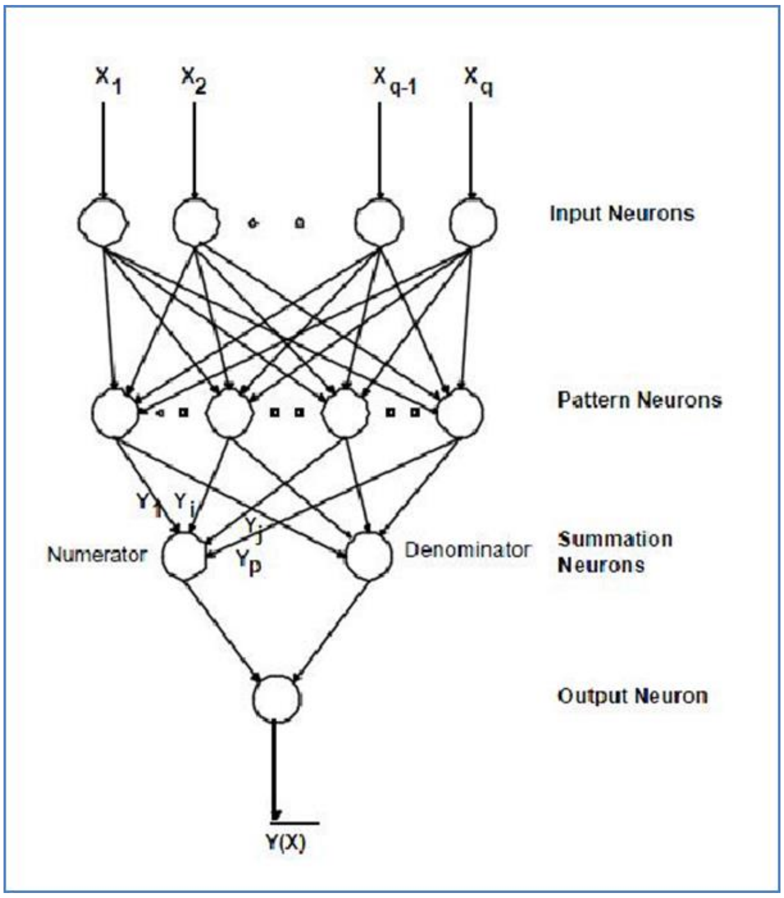

Fig. 1 The GRNN architecture

The GRNN topology consists of four layers: the input layer, the pattern layer, the summation layer, and the outputs (Fig. 1). The input layer contains a number of nodes equal to the number of input variables in the model. Each node in the input layer is fully interconnected to each node in the pattern layer. The number of pattern layer Nodes is always equal to the number of cases in the training dataset. Each node in the pattern layer is assigned a unique training vector corresponding to one of the randomly selected training set cases. The distance is calculated between each pattern layer node vector and the input layer vector, the exponent then being taken of that distance. All pattern layer nodes fully connect to the summation layer nodes. The summation layer nodes sum the values of all pattern layer nodes. There are two types of summation layer nodes: a numerator node and a denominator node. For those connections to the numerator node, the value of each pattern layer node is multiplied by the actual output value of that training case prior to summation. The predicted value is finally calculated in the output layer node by dividing the numerator node value by the denominator node value. After determining the error between actual and predicted *y* values and depending on the optimization technique used to minimize the error between those values, the above calculation may be run numerous Times with a different smoothing factor each time. Training stops once a threshold minimum error value is reached, or when the test set square error begins to rise. Equation (1) expresses how each predicted output $\hat{y}$ is a function of the corresponding output components *y* associated with each stored pattern x_i:

$$\hat{y} = \frac{\sum_{i=1}^{n}(y_i * e^{-D(X,X_i)})}{\sum_{i=1}^{n}(e^{-D(X,X_i)})} \tag{1}$$

Where $\hat{y}$ is the predicted output, y_i is the *ith* case actual output variable, $D(X,X_i)$ is calculated from (2), and *n* is the total number of cases in the training data set.

$$D(X,X_i) = \sum_{j=1}^{p}\left(\frac{X_j - X_{ij}}{\sigma_j}\right)^2 \tag{2}$$

Where *D(X, Xi)* is the distance between the input vector *x* and the *ith* case vector x_i and x_j is the *jth* data value in the input vector, x_{ij} is the *jth* data value in the *ith* case vector, and σ_j is the smoothing factor (GRNN spread) for the *jth*.