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.

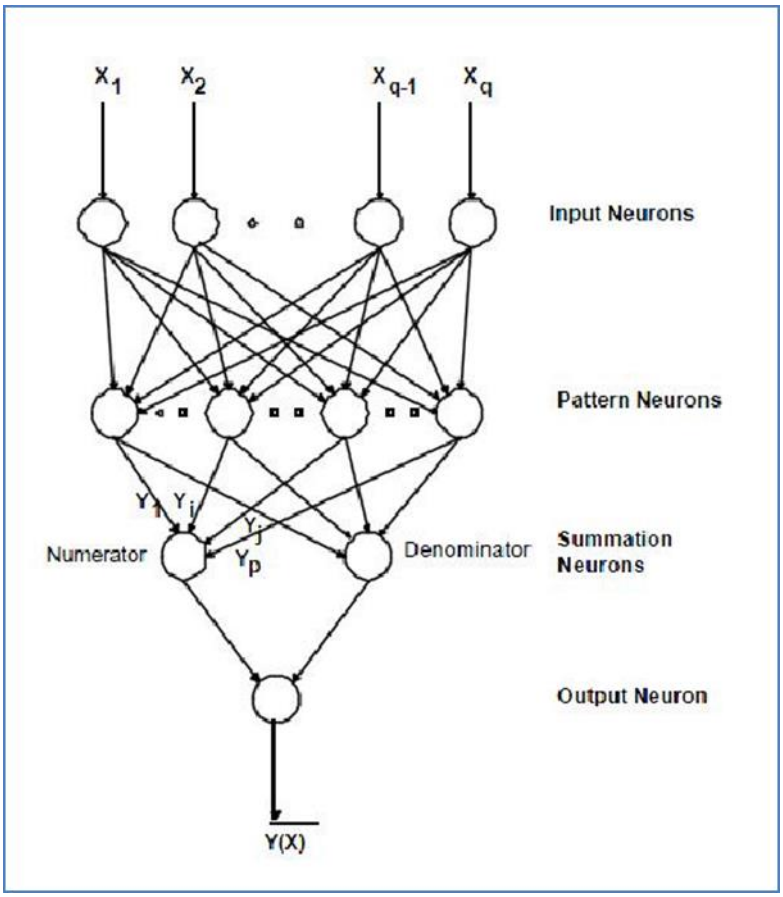

Fig. 1 The GRNN architecture

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:

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

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.

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

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*.

Specht suggests in [2] the use of the holdout method to select a good value of σ. In the holdout method, one sample of the entire set is removed and for a fixed σ, GRNN is used again to predict this sample with the reduced set of training samples. The squared difference between the predicted value of the removed training sample and the training sample itself is then calculated and stored. The removing of samples and prediction of them again for this chosen σ is repeated for each sample-vector. After finishing this process the mean of the squared differences is calculated for each run. Then the process of reducing the set of training samples and predicting the value for these samples is repeated for many different values of σ. The σ for which the sum of the mean squared difference is the minimum of all the mean squared differences is the σ that should be used for the predictions using this set of training samples. According to Specht there are no restrictions to this process, but unfortunately it turned out that for certain conditions this process does not show the desired results.

3. Particle Swarm Optimization (PSO)

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. In a PSO system, each particle is "flown" through the multidimensional search space, adjusting its position in search space according to own experience and that of neighbouring particles. A particle therefore makes use of the best position encountered by itself and that of its neighbours to position itself toward an optimal solution. The effect is that particles "fly" towards a minimum, while still searching a wide area around the best solution. The performance of each particle (i.e. the "closeness" of a particle to the global optimum) is measured using a predefined fitness function which encapsulates the characteristics of the optimization problem. Each particle i maintains the following information: x_i , the current position of the particle; v_i, the current velocity of the particle; and y_i, the personal best position of the particle. The personal best position associated with a particle i is the best position that the particle has visited so far, i.e. a position that yielded the highest fitness value for that particle. If f denotes the objective function, then the personal best of a particle at a time step t is updated as:

$$y_i(t+1) = \begin{cases} y_i(t) & if\ f(x_i(t+1)) \geq f(y_i(t)) \\ x_i(t+1) & if\ f(x_i(t+1)) < f(y_i(t)) \end{cases} \quad (3)$$

Two main approaches to PSO exist, namely *lbest* and *gbest*, where the difference is in the neighbourhood topology used to exchange experience among particles. For the *gbest* model, the best particle is determined from the entire swarm. If the position of the best particle is denoted by the vector $\hat{y}$, then:

$$\hat{y}(t) = \min\{ f(y_0(t)), f(y_1(t)), \dots, f(y_s(t)) \} \quad (4)$$

Where s is the total number of particles in the swarm. For the *lbest* model, a swarm is divided into overlapping neighbourhoods of particles. For each neighbourhood N_j , a best particle is determined with position $\hat{y}_j$. This best particle is referred to as the neighborhood best particle, defined as:

$$N_j = \{ y_{i-l}(t), y_{i-l+1}(t), \dots, y_{i-1}(t), y_i(t), y_{i+1}(t), \dots, y_{i+l}(t) \} \quad (5)$$
$$\hat{y}_j(t+1) \in N_j \mid f(\hat{y}_j(t+1)) = \min\{f(y_i)\}\ \forall\ y_i \in N_j$$

Neighbourhoods are usually determined using particles indices [7], although topological neighbourhoods have been used also [3]. The *gbes*t PSO is simply a special case of *lbest* with $l=s$; that is, the neighbourhood is the entire swarm.

For each iteration of a *gbest* PSO algorithm, v_i and x_i are updated as follows:

$$V_i(t+1) = wv_i(t) + c_1 r_1(t)(y_i(t) - x_i(t)) + c_2 r_2(t)(\hat{y}(t) - x_i(t)) \quad (6)$$
$$X_i(t+1) = x_i(t) + v_i(t+1)$$

Here w is the inertia weight, c_1 and c_2 are the acceleration constants and $r_1(t), r_2(t) \sim u(0,1)$ .Fig.2 illustrates how each of the influence components combine to result in an iterated particle velocity and subsequently position. The reader is referred to [4] for a study of the relationship between the inertia weight and the acceleration constants in order to select values which will ensure convergent behaviour. Velocity updates are also clamped to prevent them from exploding, thereby causing premature convergence. The PSO algorithm performs repeated applications of the update equations above until a specified number of iterations have been exceeded, or until velocity updates are close to zero. The quality of particles is measured using a fitness function which reflects the optimality of the corresponding solution.Fig.3 shows PSO flow diagram.

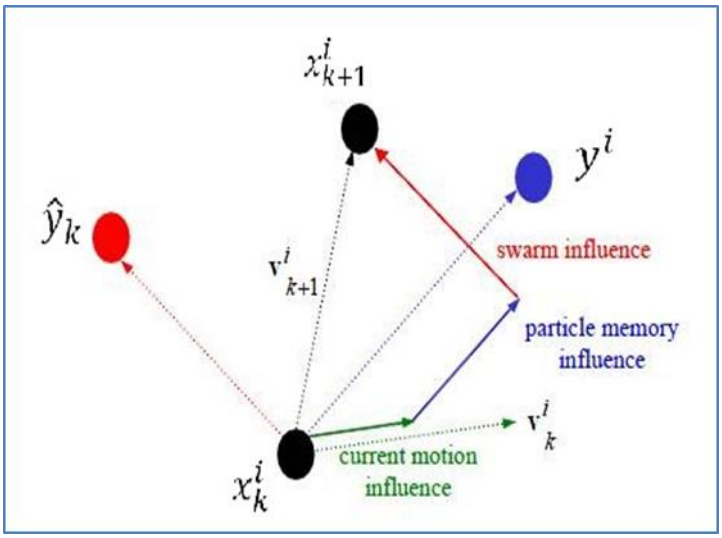

Fig. 2 Influences on particle swarm velocity iteration

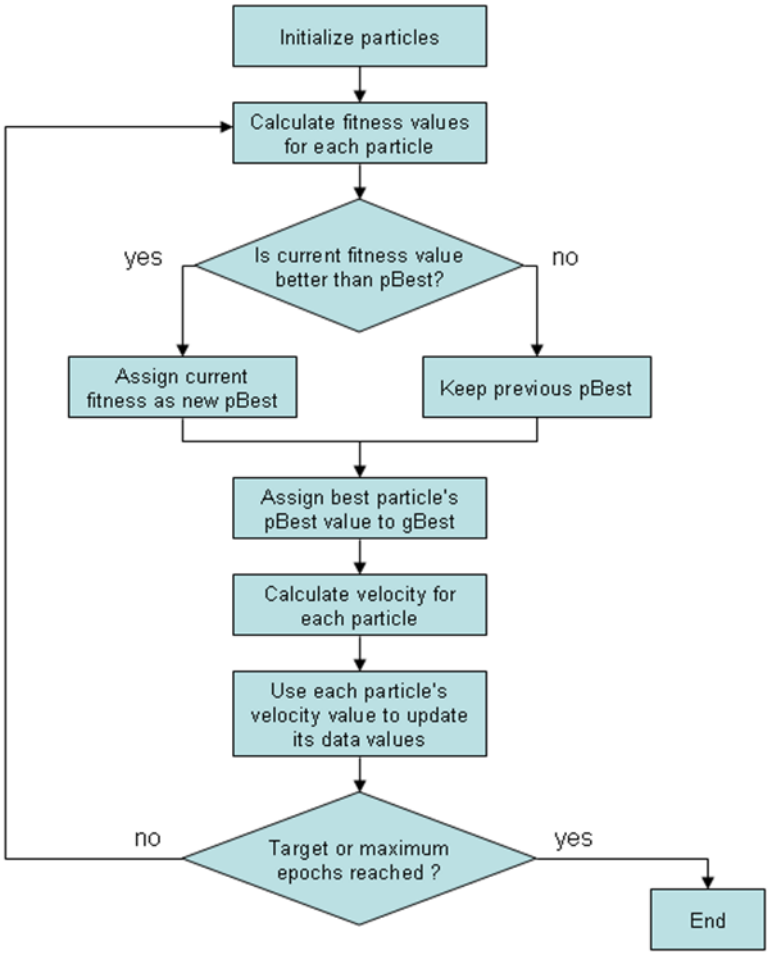

Fig. 3 Flow diagram illustrating the particle swarm optimization algorithm

4. GRNN optimization using PSO for Persian digits recognition

Digit recognition is used to verify the amount of Czech read phone numbers and postal code and etc. In this study PSO is used to optimize GRNN to recognize Persian digits. The GRNN spread value is determined by Particle Swarm Optimization.

We use images of the HODA digits dataset for the experiments. HODA digits dataset is a dataset of handwritten Farsi digits that presented with Hossein Khosravi and Ehsanollah Kabir in [13]. There are

102,352 digits in this dataset were extracted from about 12,000 registration forms of two types, filled by B.Sc. and senior high school students. The dataset presenters selected 60,000 samples for training set and 20,000 for test. The remaining samples are also available in another subset.

We also used 60000 samples for training set and 20000 for test. Optimization variable is spread value. First we scaled training and test samples to same size. Then scaled images were divided into a number of zones. Since present digits are white and their backgrounds are black, we can consider amount of white pixels in each zone as that zone feature. Because digits are handwritten, we choose a number between 0 and 255 as threshold, then consider pixels with intensity greater than that as part of digit, and consider pixels with intensity smaller than that as background pixels. Thus for each image, some features are achieved. Now we use these features as GRNN inputs.

Then we use Particle Swarm Optimization for determine best spread (σ) value for GRNN. Here the cost function is number of patterns from test set that network recognize incorrectly, and we use this cost function for determine each particle cost in PSO.

When PSO stop condition satisfied, algorithm stops. Last *gbest* value stores best GRNN spread that network that simulates with it, recognizes most Persian digits correctly.

5. Simulation AND Results

Thus above mentioned, we use images of the HODA digits dataset for the experiments. In this database, there are 102,352 images include 10 digits. These images resolution are 200 dpi and each image contains a white digit in black background.

Images in HODA have different sizes; therefore first we resize all images to 40*30 pixels and then divide each image to 5*5 zones. After that we count number of pixels that their intensity greater than a threshold (for our experiment, we consider threshold as 50).Thus for each image, 25 features are achieved and we use them as GRNN inputs.

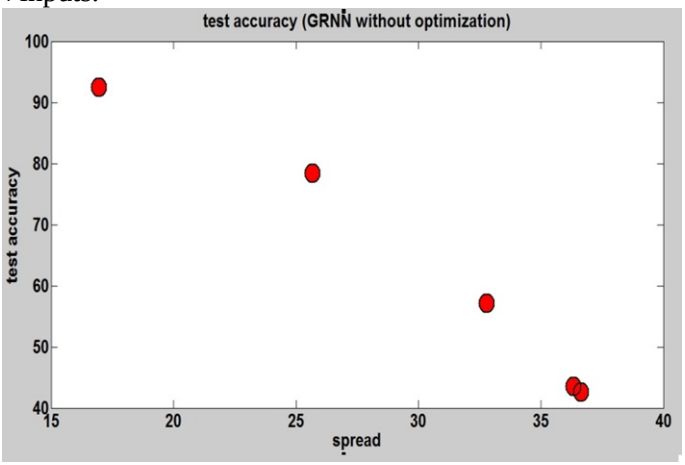

Fig. 4 accuracy of GRNN without any optimization

Then we consider 60000 training images as GRNN centers and determine a code for each digit as target value of the network .Now to determine the digit of each test image, compare that image features with centers images features via GRNN.

For GRNN simulation, we should determine its spread. For a bigger spread, the possible representation of the point of evaluation by the training sample is possible for a wider range of images. For a small value of the spread the representation is limited to a narrow range of images, respectively.

We first simulate GRNN without any optimization and determine its spread randomly.Fig.4 shows test image recognition accuracy versus spread in GRNN without any optimization.

Then we use PSO for GRNN optimization. The PSO cost function is number of images from test set that network recognize correctly; here our problem is maximization. Because of too much test data, PSO cost calculation takes long times; since we choose 1000 data from test data randomly and calculate PSO cost for them. Fig.5 shows PSO cost versus iteration in one algorithm running and Fig.6 shows PSO cost versus spread in same algorithm running. Finally used last *gbest* to GRNN spread and simulate network with this spread.Fig.7 shows test image recognition accuracy versus spread in GRNN that optimized with PSO.

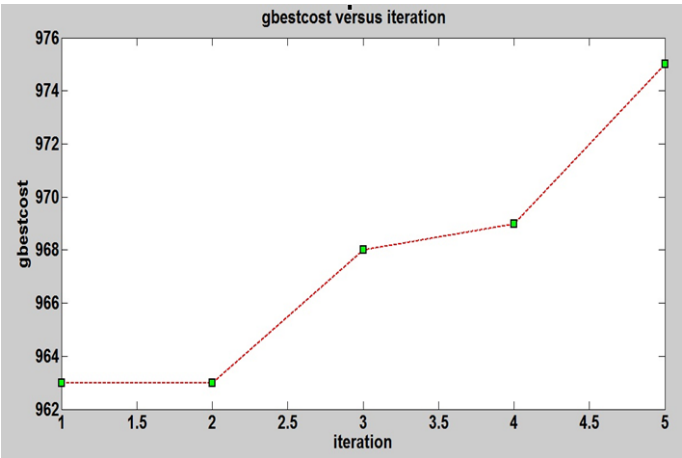

Fig. 5 gbestcost of PSO versus iteration for 1000 random test data

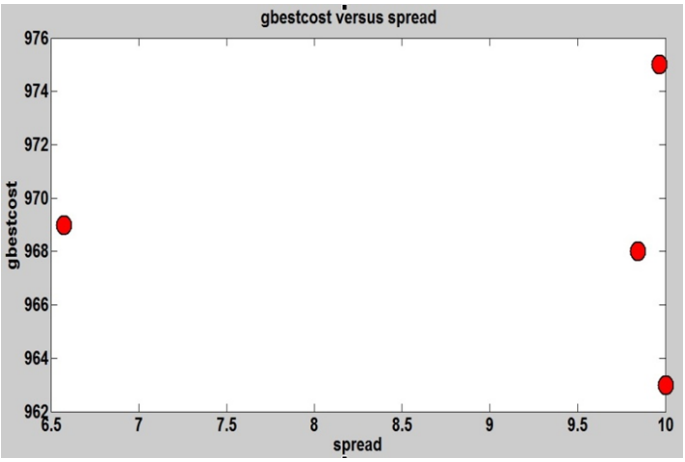

Fig. 6 gbestcost of PSO versus spread for 1000 random test data

In our proposed algorithm for test samples, 96.29% of instances were classified correctly in their classes. Table 1 shows our algorithm results and compares with GRNN without optimization results.

TABLE 1 Results of algorithm running

Run	GRNN without optimization		GRNN optimized with PSO	
	best spread	test samples accuracy	best spread	test samples accuracy
run 1	32.7742	57.18	7.9737	96.29
run 2	36.3259	43.585	6.0000	96.235
run 3	16.9525	92.54	7.0369	96.285
run 4	36.6217	42.585	6.0000	96.235
run 5	25.662	78.375	9.9648	95.965

6. Conclusion

This paper investigates the performance of GRNN optimized by Particle Swarm optimization for Persian digit recognition. We use HODA dataset for our Persian digit database. Here we used 60000 images from HODA dataset for network training and used 20000 instances for network test. Results show that the achieve network can recognize Persian digits with high accuracy.

7. References

[1] J Kennedy, RC Eberhart, Particle Swarm Optimization, Proceedings of the IEEE International conference on Neural Networks, Vol 4, Perth, Australia, pp 1942-1948, 1995.

[2] Specht, D. F., A general regression neural network. IEEE Transactions on Neural Networks,(1991), 568–576.

[3] PN Suganthan, Particle Swarm Optimizer with Neighborhood Optimizer, Proceedings of the Congress on Evolutionary Computation, pp 1958-1962, 1999.

[4] F van den Bergh, An Analysis of Particle Swarm Optimizers, PhD Thesis, University of Pretoria, South Africa, 2002.

[5] J Kennedy, RC Eberhart, Particle Swarm Optimization, Proceedings of the IEEE International Conference on Neural Networks, Vol 4, Perth, Australia, pp 1942-1948, 1995.

[6] J Kennedy, RC Eberhart, Swarm Intelligence,Morgan Kaufmann,2001.

[7] Y Shi, RC Eberhart, Parameter Selection in Particle Swarm Optimization, Evolutionary Programming, Vol VII, Proceedings of Evolutionary Programming, pp 591-600, 1998.

[8] M Omran, A Salman, AP Engelbrecht, Image classification using Particle Swarm Optimization, Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning,2002

[9] Polat, O. Yildirim, T. (2007). Genetic optimization of GRNN for pattern recognition without feature extraction, Elsevier, Expert Systems with Applications 34 (2008) 2444–2448.

[10] Polat, O. Yildirim, T.(2006).Hand geometry identification without feature extraction by general regression neural network, Elsevier Expert Systems with Applications 34 (2008) 845–849.

[11] Currit, N.(2002).Inductive regression: overcoming OLS limitations with the general regression neural network, Elsevier Computers, Environment and Urban Systems, 26 (2002) 335–353.

[12] Chen, A. Leung, M.(2004). Regression neural network for error correction in foreign exchange forecasting and trading, Elsevier Computers & Operations Research 31 (2004) 1049–1068.

[13] Hossein Khosravi, Ehsanollah Kabir. Introducing a very large dataset of handwritten Farsi digits and a study on their varieties, Pattern Recognition Letters 28 (2007) 1133–1141.