Characters and Digits Recognition Using Neural Network Learned by Particle Swarm Optimization

Khamis A. Yousif
M. Sc. in Computer Science/Public Health Dept./College of Veterinary Medicine/Diyala University
Email: alkasi_68@yahoo.com

Abstract

The meaning of the Particle Swarm Optimization (PSO) refers to a relatively new family of algorithms that may be used to find optimal (or near optimal) solutions to numerical and qualitative problems. Neural Network is an information processing system that has been developed as generalization models of human cognition of neural biology.

In this paper the neural network learned by PSO method to solve one of pattern recognition problems which is considered as one of the important applications in the classification field, instead of using Back Propagation (BP) or Genetic Algorithm (GA) methods. The suggested method is found to learn the NN, to solve characters and digits or decimal numbers (0..9) recognition problem, by modifying the NN weights, this is done by calculating the fitness value which is considered as a threshold value. A comparison studies are made between PSO and Back Propagation (BP) methods in NN learning to specify which is better in solving letter recognition problem.

Keywords: Neural Network, Artificial Neural Network, Particle Swarm Optimization, Pattern Recognition, Back Propagation, Classification field.

Characters and Digits Recognition Using Neural Network Learned by Particle Swarm Optimization

Khamis A. Yousif

تمييز الأحرف والأرقام باستخدام الشبكات العصبية المعلمة بواسطة سرب الجسيمات الأمثل

م.م خميس احمد يوسف

مستخلص

أن تحقيق أمثلية السرب الجزيئي تعني الآشارة الى عائلة جديدة من الخوارزميات التي تستخدم لايجاد حلول مثالية (أو أقرب الى المثالية) للمسائل العددية والكمية. الشبكات العصبية هي نظام معالجة البيانات الذي طور كنموذج لتعميم الادراك البشري لعلم الاحياء البايلوجية.

في هذا البحث تم تعليم الشبكة العصبية بطرقة أمثلية السرب الجزيئي لحل احد مسائل تمييز الانماط، بدلا من طريقة الانتشار التراجعي او الخوارزمية الجينية. الطريقة المقترحة وجدت لتعليم الشبكة العصلبة، لحل مسألة تمييز الرموز والارقام العشرية (0..9)، من خلال تعديل الاوزان، وهذا تم من خلال حساب قيمة الأفضلية والتي اعتبرت هي قيمة العتبة. وايضا تم اجراء دراسة مقارنة بين طريقة أمثلية السرب الجزيئي والانتشار التراجعي عند تعليم الشبكة لتحديد الطريقة الافضل عند حل مسألة تمييز الحروف.

مفتاح الكلمات: *الشبكات العصبية , الشبكات العصبية المصنعة , سرب الجسيمات الأمثل , تمييز الأنماط , مجال التصنيف.*

Introduction

A ***Neural Network*** (NN) is not programmed to solve a problem-instead, it learns to solve a problem [1]. A NN may benefit from the application of solving methods to its training. One of the important new learning methods is a ***Particle Swarm Optimization*** (PSO), which is simple in concept, has few parameters to adjust and easy to implement. PSO has found applications in a lot of areas. In general, all the application areas that the other evolutionary techniques are good at are good application areas for PSO [2].

In 1995, Kennedy J. and Eberhart R. [3], introduced a concept for the optimization of nonlinear functions using particle swarm methodology. The evolution of several paradigms outlined, and an implementation of one of the paradigms had been discussed.

In 1999, Eberhart R.C. and Hu X. [4], arranged a new method for the analysis of human tremor using PSO which is used to evolve a NN that distinguishes between normal subject and those with tremor.

In 2004, Shi Y. [2], surveyed the research and development of PSO in five categories: algorithms, topology, parameters, hybrid PSO algorithms, and applications. There are certainly other research works on PSO which are not included due to the space limitation.

Artificial Neural Network

Artificial Neural Network (ANN) is an information-processing system that has certain performance characteristics in common with biological neural networks. ANNs represent an important area of research, which opens a variety of new possibilities in different fields including classification or pattern recognition or predictions.

It is known by NN that can approximate functions and mathematical operators arbitrarily as well as the number of neurons in the network tends to infinity. In this respect Feed Forward Artificial Neural Networks (FFANNs) can be considered as "universal approximations" which is capable of describing the input-output relationships of mechanical systems [5].

With classification in application domain of ANNs, we have a long list of researches and real applications include speed processing, image processing and computer vision, pattern classification and recognition, system control, robotics, forecasting and modeling, optimization and management of information and medical diagnosis [6]. Also there is a great role of NNs as a means for implementing expert systems, because of their ability to solve a specific problem, producing it as if it were a black-box solution where the mode of producing answers is not clearly understood. Due to its adaptive and parallel processing ability, it has many applications in the engineering field [7].

NNs can be grouped into six areas of applications: prediction, pattern recognition, associative memories, classification, optimization and general mapping.

Model of a Neuron

A neuron is information-processing unit that is fundamental to the operation of a neural network. Figure (1) shows the block diagram of the model of a neuron, which forms the basis for designing (artificial) NN. The basic elements are identified [8]:

1. A set of synapses or connecting links, each of which is characterized by a weight or strength of its own. Specifically, a signal (x_i) $1 \leq i \leq n$ at the input of synapse (i) connected to neuron (j) is multiplied by the synaptic weight (w_{ij}) $1 \leq j \leq m$.
2. An adder for summing input signals weighted by the respective synapses of neuron.

3. An activation function for limiting the amplitude of the output of a neuron. Typically, the normalized amplitude range of the output of a neuron is written as the closed unit interval [0,1] or alternatively [-1,1].

The neuronal model is including an externally applied bias (b_k), which it has the effect of increasing or lowering the net put of the activation function. In mathematical terms:

$$Y_j = f\left(\sum_{j=1}^{n} w_{ij} x_i + b_j\right) \qquad \text{... (1)}$$

Where f (w, x, b) is the activation function.

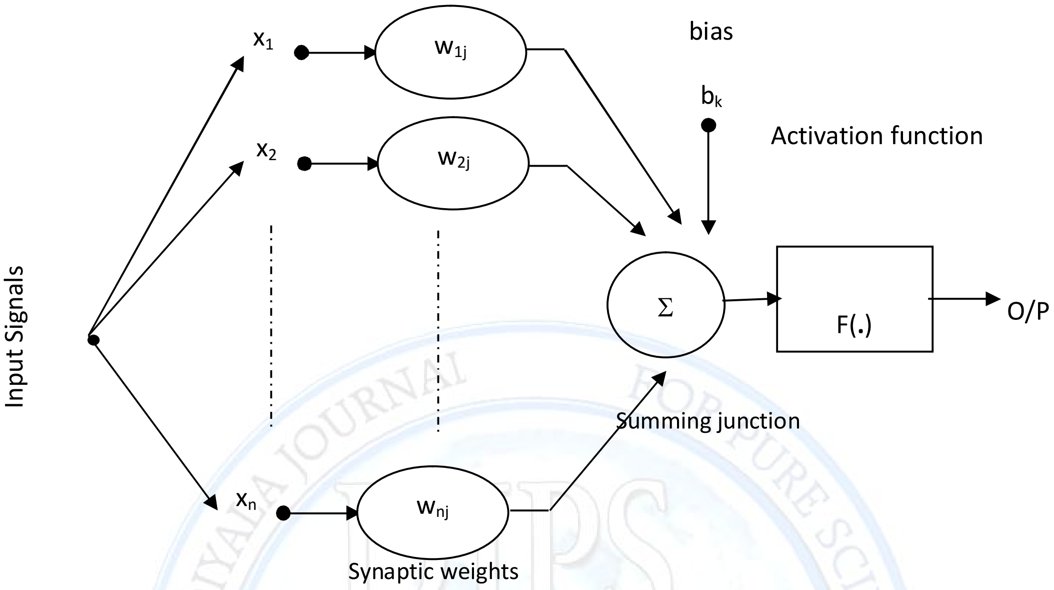

Figure (1) block diagram of model of a neuron

The output of any neuron is the result of thresholding (if any) of its internal activation, which represents the weighted sum of the neuron's inputs. The sigmoid function is by far the most common form of activation function used in the construction of ANN [9]:

$$f(x) = \frac{1}{1+e^{-\lambda x}} \qquad \dots (2)$$

Here the output values are within (0,1), where λ is the positive value.

In general, we may identify three fundamentally different classes of network architectures. To characterize a given ANN, it is necessary to specify the number of neurons, and how they are interconnected and the processing that takes place throughout the network. The multi layers feedforward networks (MLFFNs) have layer(s) of nodes (neurons) between the input and the output nodes, which is called the hidden layers [10].

Neural Network Learning Algorithms

The learning algorithm is a procedure for modifying the weights on the connection links in a neural net, and also known as training algorithms or learning rules [10].

NNs are trained by two main types of learning algorithms: **supervised** learning algorithms and **unsupervised** learning algorithms.

Unsupervised learning algorithms do not require the unknown desired outputs. A supervised learning algorithm adjusts the strengths or weights of the inter-neuron connections according to the difference between the desired and actual network outputs corresponding with a given input. The most common examples of supervised learning algorithms include Back propagation algorithm (BP) and Genetic Algorithm (GA) [9].

Particle Swarm Optimization (PSO)

PSO was originally developed by a social-psychologist J. Kennedy and an electrical engineer R. Eberhart in 1995 and emerged from earlier experiments with algorithms that modeled the “flocking behavior” seen in many species of birds. Where birds are attracted to a roosting area in simulations they would begin by flying around with no particular destination and in spontaneously formed flocks until one of the birds flew over the roosting area. PSO has been an increasingly hot topic in the area of computational intelligence. It is yet another optimization algorithm that falls under the soft computing umbrella that covers genetic and evolutionary computing algorithms as well [11].

Fitness Criterion

One of these stopping criterions is the fitness value. Since the PSO algorithm is chosen to be a supervised learning algorithm, then there are observed values of (t_i) and desired output values of (f_i). These two values have to be compared, if they are closed to each other then the fitness is good, else the algorithm must continue its calculations until this condition is satisfied or the specified number of iterations is finished.

The corrections to the weights are selected to minimize the residual error between t_i and f_i output. The Mean Squared Error (MSE) is one solution for the comparison process:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (t_i - f_i)^2 \qquad \text{… (3)}$$

Where n is the number of the compared categories.

PSO Algorithm

The PSO algorithm depends in its implementation in the following two relations:

$$v_{id}=w*v_{id}+c_1*r_1*(p_{id}-x_{id})+c_2*r_2*(p_{gd}-_{id}) \qquad \text{...(4-a)}$$

$$x_{id} = x_{id} + v_{id} \qquad \text{…(4-b)}$$

where c_1 and c_2 are positive constants, r_1 and r_2 are random function in the range [0,1], $x_i=(x_{i1},x_{i2},\dots,x_{id})$ represents the i^{th} particle; $p_i=(p_{i1},p_{i2},\dots,p_{id})$ represents the best previous position (the position giving the best fitness value) of the i^{th} particle; the symbol g represents the index of the best particle among all the particles in the population, $v=(v_{i1},v_{i2},\dots,v_{id})$ represents the rate of the position change (velocity) for particle i [2].

The original procedure for implementing PSO is as follows:

1. Initialize a population of particles with random positions and velocities on d-dimensions in the problem space.
2. PSO operation includes:
   a. For each particle, evaluate the desired optimization fitness function in d variables.
   b. Compare particle's fitness evaluation with its pbest. If current value is better than pbest, then set pbest equal to the current value, and p_i equals to the current location x_i.
   c. Identify the particle in the neighborhood with the best success so far, and assign it index to the variable g.
   d. Change the velocity and position of the particle according to equation (4a) and (4b).
3. Loop to step (2) until a criterion is met.

Like the other evolutionary algorithms, a PSO algorithm is a population based on search algorithm with random initialization, and there is an interaction among population members. Unlike the other evolutionary algorithms, in PSO, each particle flies through the solution space, and has the ability to remember its previous best position, survives from generation to another. The flow chart of PSO algorithm is shown in figure (2) [12].

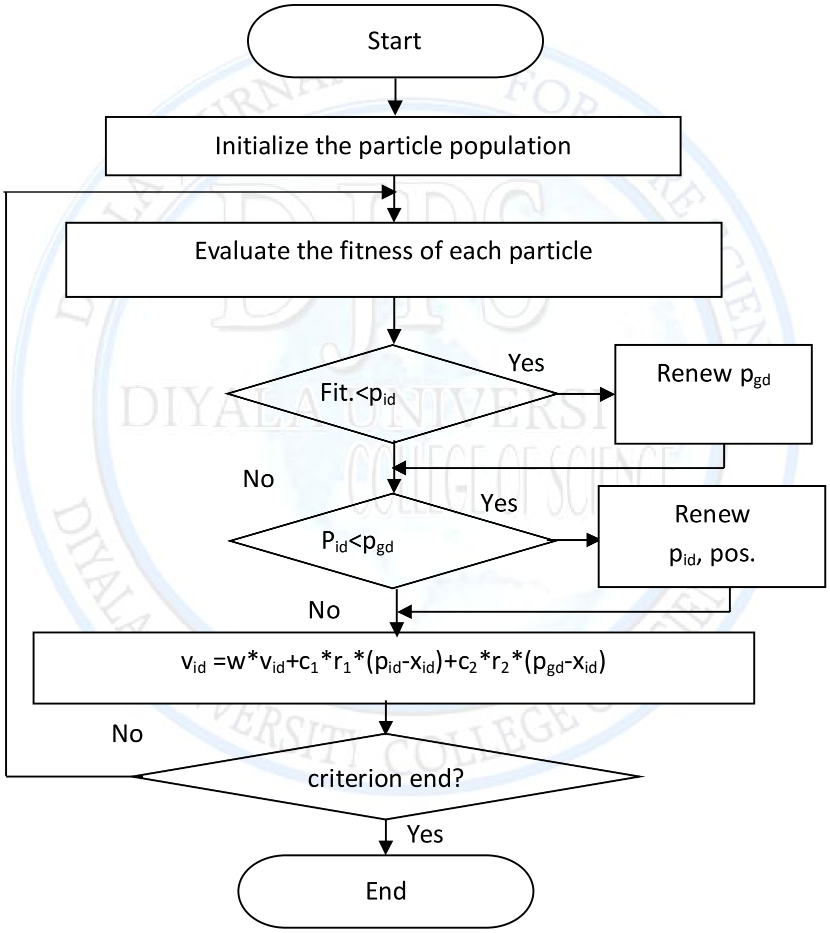
Start
Initialize the particle population
Evaluate the fitness of each particle
Fit.<p_{id}
Yes
Renew p_{gd}
No
P_{id}<p_{gd}
Yes
Renew
p_{id}, pos.
No
$v_{id} = w*v_{id}+c_1*r_1*(p_{id}-x_{id})+c_2*r_2*(p_{gd}-x_{id})$
No
criterion end?
Yes
End

Figure (2) Flowchart of PSO Algorithm [12].

The Parameters of PSO [13]

A number of factors will affect the performance of the PSO. These factors are called PSO parameters, these parameters are:

1. Number of particles in the swarm affects the run-time significantly, thus a balance between variety (more particles) and speed (less particles) must be sought.
2. Maximum velocity (v_{max}) parameter. This parameter limits the maximum jump that a particle can make in one step.
3. The role of the inertia weight w, in equation (4a), is considered critical for the PSO's convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current one.
4. The parameters c_1 and c_2, in equation (4a), are not critical for PSO's convergence. However, proper fine-tuning may result in faster convergence and alleviation of local minima, c_1 than a social parameter c_2 but with $c_1 + c_2 = 4$.
5. The parameters r_1 and r_2 are used to maintain the diversity of the population, and they are uniformly distributed in the range [0,1].

Training the ANN by using PSO

The PSO algorithm is vastly different than any of the traditional methods of training. PSO does not just train one network, but rather training networks. PSO builds a set number of ANN and initializes all network weights to random values and starts training each one. On each pass through a data set, PSO compares each network's fitness. The network with the highest fitness is considered the global best [14].

Each neuron contains a position and velocity. The position corresponds to the weight of a neuron. The velocity is used to update the weight. If a neuron is further away then it will adjust its weight more than a neuron that is closer to the global best [13].

As usual the number of interconnections (IC) for NN can be calculated by the following relations for one hidden layer:

$$IC = n * m + m*k \qquad \dots(5)$$

Where n is the number of nodes in the input layer, m is the number of nodes in the hidden layer, while k is the number of nodes in the output layer.

Particle Swarm Optimization Implementation

PSO is an extremely simple concept, and can be implemented without complex data structure. No complex or costly mathematical functions are used, and it doesn't require a great amount of memory [1]. The facts of PSO has fast convergence, only a small number of control parameters, very simple computations, good performance on neural networks, and the lack of derivative computations made it an attractive option for training the NN.

In this paper, the binary PSO was implemented in classification applications. From the classification application, the letter recognition problem is chosen.

A comparative study is made on the computational requirements of the PSO and BP as a training algorithm for NN's.

A NN-Learning system has been proposed to solve the mentioned problems by Binary PSO (BPSO) and BP algorithms.

BPSO Algorithm

The original procedure of binary PSO algorithm is as follows:

1. Read NN information file.
2. Change (Integer or real) input data to binary data.
3. Initialize a population of particles with random positions and velocities on d-dimensions in the problem space.
4. PSO operations.
5. Loop to step (4) until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations.

NN-Learning System Implementation

The proposed system called NN-learning system since it can be applied on BP algorithm as well as on PSO algorithm which applied on pattern recognition problem. The block diagram of NN-learning system implementation shown in figure (3).

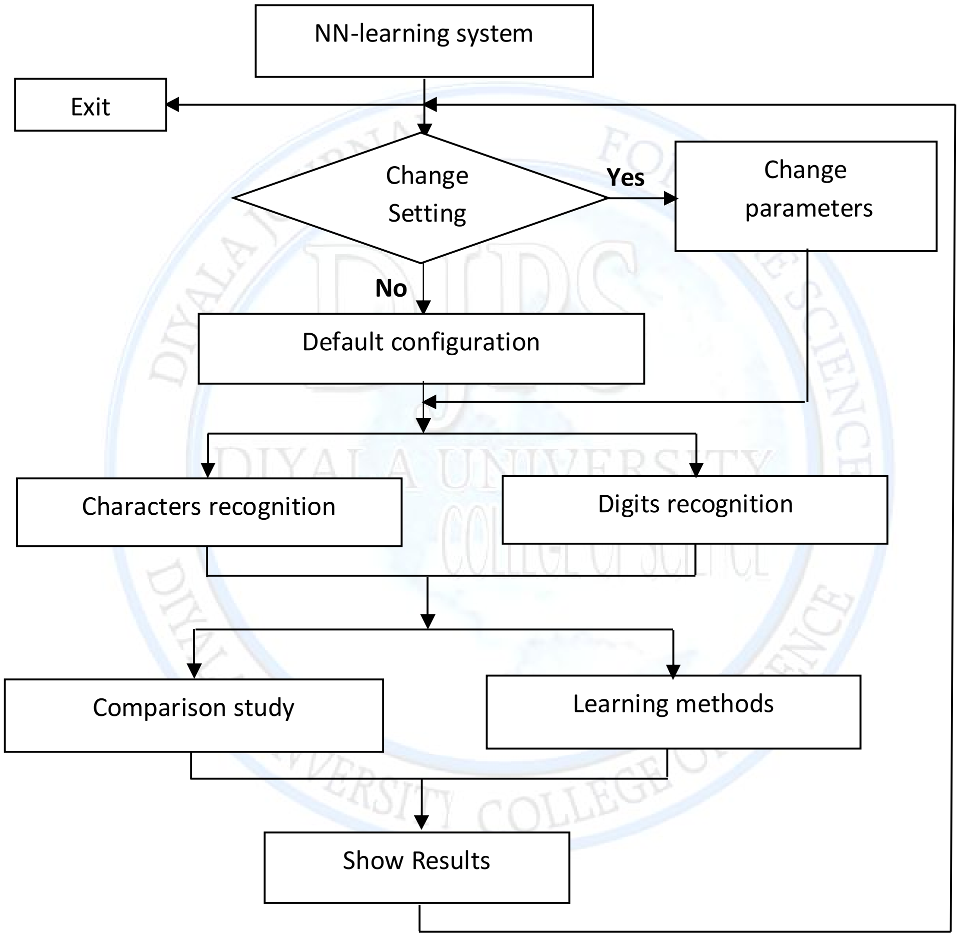

Figure (3) Block diagram of NN-Learning system

Pattern Recognition Applications

One of the most important applications of NN is the classification problems which are divided into many fields; our interest of classification problem is the Pattern Recognition (PR) problems.

Pattern Recognition is the research area that studies the operation and design of systems that recognize patterns in data.

PR aims to classify data (patterns) based on either a priori knowledge or on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations, defining points in an appropriate multidimensional space. Important application areas are image analysis, speech analysis, man and machine diagnostics, person identification and industrial inspection [15].

Binary PSO algorithm has been used to recognize patterns of character and decimal number. The network is initially learned upon symbols and tested on another set of symbols to illustrate its efficiency in recognizing new pattern.

PR-Neural Network Construction

In this paper, the patterns are represented by d*e matrix. The data of the matrix consists of 1’s and 0’s only. The objects of interest are the 1’s and the background consists of 0’s. The output symbols consist of k bits which must cover all the possible patterns that mean 2^k possible, these 2^k is the targets. The d*e binary values are stored in a file in a one dimensional array row by row arrangement. In this manner since the number of output nodes k>1, then the MSE value, must be rewritten as in relation (6).

$$\text{MSE}=\frac{1}{Ns*k}\sum_{i=1}^{Ns}\sum_{j=1}^{k}(t_{ij}-f_{ij})^2 \qquad \dots (6)$$

Where Ns is the number of training set.

The proposed LR neural network consists of 3 layers, with (d, m, k) neurons which represent the number of neuron in input, hidden and output layers respectively. The proposed PR-NN is shown in figure (4).

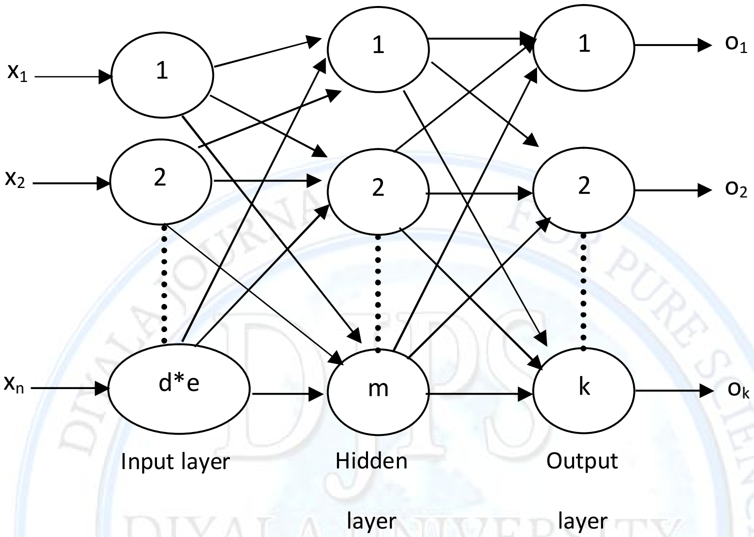

Figure (4) the proposed PR-NN (d*e,m,k).

PSO Implementation in PR-NN

Two examples of pattern recognitions are used to be implemented by PSO. First it has to choose random weights for the network interconnections ranging between 0 and 1. We need to use the following symbols in the tables of PSO implementation:

- **Exp:** Experiment number.
- **NoI**: Number of Iterations.
- **MSE**: least fitness (Mean Square Error).
- **AL:** Accuracy Level of the learning.

1- Character Recognition (CR)

In this section, PSO neural network is used to recognize a number of different symbols. We used 5 types of patterns to be learned. Every pattern is represented by 5*5 array (d=5, e=5), this means that 25 neurons in the input layer. Figure (5) shows the five character input matrices.

0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	1	0	0	0
1	0	0	0	0

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

0	0	0	0	0
0	0	0	0	0
1	1	1	1	1
0	0	0	0	0
0	0	0	0	0

0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0
0	0	1	0	0

0	1	1	1	0
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
0	1	1	1	0

Figure (5) character input matrices.

The output data of every pattern can be covered by 3 bits, this means that the output layer consists of 3 neurons and the training set Ns=5. The output binary coding of CR is shown in table (1).

Table (1) the output binary coding of CR.

Pattern	Shape	Output
Italic line from right to left	╱	001
Italic line from left to right	╲	010
Horizontal line	—	011
Vertical line	|	100
Circle	○	101

In the hidden layer we suggest to use 7 neurons; the results of PSO implementation in (25-7-3) CR-NN for 3 experiments are shown in table (2).

Table (2) PSO implementation in (25-7-3) CR-NN.

Experiments	NoI	MSE	AL
1	106	0.0094260	100.00%
2	97	0.0098676	100.00%
3	96	0.009846	100.00%

2- Decimal Number Recognition (DR)

In this research, we used (10) patterns to be learned, these patterns represent the decimal number from 0 to 9. Every pattern is represented by 6*5 (d=6, e=5) array this means that 30 neurons in the input layer, for example if the number is "4" which is the 5th number in the decimal numbers set, then the output data is "0100", since the output layer consists of 4 neurons and the training set Ns=10. The matrices data of the decimal numbers (0..9) are shown in figure (6).

	0					1					2					3					4				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	0	0	1	0	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	1	1	0
2	0	1	0	1	0	0	1	1	0	0	1	0	0	1	0	1	0	0	1	0	0	1	0	1	0
3	1	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	0
4	1	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0	1	1	1	1	0
5	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1	0
6	0	0	1	0	0	0	1	1	1	0	1	1	1	1	1	0	1	1	0	0	0	0	1	1	1

	5					6					7					8					9				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	0	1	1	1	1	0	0	1	1	0	1	1	1	1	1	0	0	1	1	0	0	0	1	1	0
2	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	0	1	0	0	1	0	1	0	0	1
3	0	1	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	0	0	1
4	0	0	0	0	1	0	1	1	1	1	0	0	1	0	0	0	1	0	0	1	0	0	1	1	1
5	0	1	0	0	1	0	1	0	0	1	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1
6	0	0	1	1	0	0	1	1	1	1	1	0	0	0	0	0	0	1	1	0	0	1	1	1	1

Figure (6) Input matrix data of the decimal numbers.

In the hidden layer we suggest to use 10 neurons.

The results of PSO implementation in (30-10-4) DR-NN for 3 experiments shown in table (3).

Table (3) PSO implementation in (30-10-4) DR-NN.

Experiment s	**NoI**	**MSE**	**AL %**
1	1182	0.009986	95.00%
2	797	0.009886	97.50%
3	1049	0.009990	96.00%

Comparison Results between PSO and BP

The multi-layer perceptrons established for relevant purposes are trained with PSO algorithm and BP algorithm, respectively, through the selected training set. In order to establish a fair start-up state for BP-based perceptrons, the training processes of BP-based perceptrons always start with best solution in the initial population used for the processes of the counterparts, PSO-based perceptrons. For BP-based perceptrons, such “evolution time” directly equals the times of its updated iterations, while for PSO-based perceptrons,

it is equivalent to a production of the population size and the evolving generations. In order to compare performances between PSO and BP-based perceptrons, the training histories are recorded with the same fitness evolution times for both perceptrons to be compared with.

In this study, some examples of character and decimal number recognition are chosen, in order to make a comparison study for PR-NN between PSO and BP algorithms.

In the PR-problems we chose the same two problems discussed before, these problems are CR with Ns=5 and for DR with Ns=10, for our comparative study between PSO and BP. The implementation is done for 3 experiments for each problem. Table (4) and (5) show the comparison results for two PR problems.

Table (4) comparison results between PSO and BP [16] for CR-NN.

Experiment	Algorithm	NoI	AL%
1	PSO	101	100.00%
	BP	3113	80.00%
2	PSO	162	100.00%
	BP	2542	92.00%
3	PSO	120	100.00%
	BP	3946	84.00%

In [16], the researchers solving CR by BP algorithm, its obvious that PSO is butter than BP because of the thousands of iterations and bad accuracy level as shown in table (4). It's important to mentioned that the results of BP implementation in this problem taken from [16].

Table (5) comparison results between PSO and BP for DR-NN.

Experiment	**Algorithm**	**NoI**	**MSE**	**AL%**
1	PSO	1436	0.0098014	97.50%
	BP	1682	0.0099700	97.50%
2	PSO	1213	0.0099566	98.00%
	BP	1604	0.0099000	96.00%
3	PSO	1072	0.0099550	97.50%
	BP	1570	0.0100000	96.50%

Conclusions

This paper concludes the following aspects:

1. The PSO algorithm has fewer parameters to tune, thus it is more universal tool, and can be used to locate or track stationary as well as non-stationary extremes.
2. The BP algorithm, however, is a local search method. It is easily falling into local minima and fails to find the global optimum when used to train a perceptron, while PSO algorithm, in spite of its population stochastic search method, it is less in falling in local minima and it can find global weights with both large probability and fast convergence rate during the training of NN.
3. The binary PSO-based perceptrons perform better than the BP-based perceptrons in application of pattern recognition.
4. A hybrid can be made between a PSO algorithm and any other learning NN algorithm to enhance or develop the learning methods achievement.
5. A comparison study between PSO algorithm and GA can be made to be evaluated which is better on this application.

References

1. Clow B. “A Comparison of Neural Network Training Methods for Character Recognition”, Department of Computer Science Carleton University, 95.495, 2003.
2. Shi Y., “Particle Swarm Optimization”, Electronic Data Systems, Inc. Kokomo, IN 46902, USA Feature Article, IEEE Neural Networks Society, February 2004.
3. Kennedy J. and Eberhart R. C. “Particle Swarm Optimization”, Proceedings of IEEE International Conference on NN, Piscataway, pp. 1942-1948, 1995.
4. Eberhart R. C. and Hu X. “Human Tremor Analysis Using Particle Swarm Optimization” Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999), Washington D.C. pp. 1927-1930, 1999.
5. Eskandarian A., and et al, “Vehicle Crash Modeling Using Recurrent Neural Networks”, Math. & Computer Modeling vol. 28, No. 9, pp. 31 - 42, 1998.

6. Haykin S., “Neural Network–a Comprehensive Foundation”, prentice Hall, 1999.
7. Zurada J. M., “Introduction to Artificial Neural Network”, Jaico Publishing House, 1996.
8. Faiz A. “Neuro-Genetic Based Switching Controller for Optimal Inverters”, Ph. D. thesis, University of Technology, Baghdad, 1998.
9. Chen C., “Classification of Under Water Signals Using Wavelet Transforms and Neural Networks”, Math. & computer Modeling, Vol.22, No2, pp. 47-60, 1998.
10. Amin, S.M. and Rodin E. Y., “Neurocontrol of Nonlinear System via Local Memory Neurons”, Math. & computer Modeling vol. 27, No. 3, pp. 65-92, 1998.
11. Pomeroy P. “An Introduction to Particle Swarm Optimization”, Article, Mar, www.adaptiveview.com, pages. 1-7, 2003.
12. Zhou Y., and et al, “Particle Swarm Optimization Based Approach for Optical Finite Impulse Response Filter Design”, Optical Society of America, 2003.
13. Settles M. and Rylander B., “Neural Network Learning using Particle Swarm Optimizers”, Advances in Information Science and Soft Computing, pp. 224-226, 2002.
14. Hu X., Eberhart R.C. and Shi Y., “Swarm Intelligence for Permutation Optimization: A case Study of n-Queens Problem”, Indiana, USA, 2003.
15. Denoeux T., “Pattern Recognition for System Monitoring - An Overview”, University of Compiègne CNRS (National Center for Scientific Research), France, Oct., 2002.
16. Riadh A.K., Jubair J., Dina N.A., “Using Neural Networks In Pattern Recognition”, مجلة ابحاث الحاسوب، المجلد الرابع-العدد الثاني، العراق , 200