Study of Principle Component Analysis and Learning Vector Quantization Genetic Neural Networks

Dr. Mazin Z. Othman* **& Dr. Arif A. Al-Qassar****

Received on: 10/4/2008
Accepted on: 4/12/2008

Abstract

In this work, the Genetic Algorithm (GA) is used to improve the performance of Learning Vector Quantization Neural Network (LVQ-NN), simulation results show that the GA algorithm works well in pattern recognition field and it converges much faster than conventional competitive algorithm. Signature recognition system using LVQ-NN trained with the competitive algorithm or genetic algorithm is proposed. This scheme utilizes invariant moments adopted for extracting feature vectors as a preprocessing of patterns and a single layer neural network (LVQ-NN) for pattern classification. A very good result has been achieved using GA in this system. Moreover, the Principle Component Analysis Neural Network (PCA-NN) which its learning technique is classified as unsupervised learning is also enhanced by hybridization with the genetic algorithm. Three algorithms were used to train the PCA-NN. These are Generalized Hebbian Algorithm (GHA), proposed Genetic Algorithm and proposed Hybrid Neural/Genetic Algorithm (HNGA).

Keywords: Neural Networks, Genetic Algorithms, Principle Component Analysis, Learning Vector Quantization, Signature Recognition.

دراسة الشبكات العصبية الجينية والمعتمدة على أساس تحليل المركبات الأساسية والشبكات ذات التعليم ألاتجاهي الكمي

الخلاصة

في هذا البحث تم استخدام الخوارزميات الجينية لتحسين اداء الشبكة العصبية ذات التعليم الاتجاهي الكمي . النتائج التمثيلية كانت جيدة في مضمار استطلاع البيانات وانها تتوصل الى النتائج بصورة اسرع . تم استخدام خوارزميات المنافسة بطريقة العزوم الثابتة للحصول على متجة الصفات كخطوة اولى في تعليم الشبكة الاحادية الطبقة والمعتمدة في تعلمها على الخورزميات الجينية. علاوة على ذلك تم استخدام الشبكة العصبية التي تعتمد على اساس تحليل المركبات الاساسية في تعليم الشبكة بدون مشرف. ثلاثة خوارزميات تم استخدامها وهي الخوارزمية الهيبية العامة واخرى مقترحة ومسندة بالخوارزميات الجينية والخوارزمية المدمجة بين الشبكات العصبية والخوارزميات الجينية.

1. Introduction

In the past decade, two areas of research, which have become very popular, they are the fields of Neural Networks (NNs) and Genetic Algorithms (GAs). Both are computational abstractions of biological information processing systems, and both have captured the imaginations of researchers all over the world. In general, NNs are used as learning systems and GAs as optimization systems; but as many researchers have discovered, they may be combined in a number of different ways resulting in highly successful adaptive systems [1].

Many papers are recently published that utilize neural networks in the field of signature recognition [2], [3], [4]. On the other hand , the genetic algorithm is also encouraged the researchers to use it in the signature recognition as a tool to select the features that best define the

*Head of Electrical and Electronic Engineering Dept., Gulf University- Kingdom of Bahrain
** Control and Systems Engineering Department, University of Technology / Baghdad

signature [5] or to minimize the difference between the comparable signatures in the field of signature verification [6].

In the current work, a special neural network structures, mainly the learning vector quantization (LVQ) and principal component analysis (PCA) have been learned using genetic algorithms in order to have a best performance from both techniques.

2. Learning Vector Quantization Neural Networks (LVQ-NN)

The learning vector quantization uses competitive learning rules in an attempt to define decision boundaries in the input space. The networks which perform LVQ are trained with supervision (because these networks are given a set of input patterns along with correct class labels). A competitive layer automatically learns to classify input vectors, depend on the distance between input vectors, if two input vectors are very similar, the competitive layer probably will put them in the same class.

LVQ networks, on the other hand, learn to classify input vectors into target classes chosen by the user. After training, an LVQ network classifies an input vector by assigning it to the same class as the output unit that has its weight vector closest to the input vector [7].

2.1 The Structure of LVQ-NN

A neural network for learning vector quantization consists of two layers: an input layer and an output layer as shown in Fig. (1). It represents a set of reference vectors, the coordinates of which are the weights of the connections leading from the input neurons to the output neuron, hence, one may also say that each output neuron corresponds to one reference vector [7].

2.2 Competitive Learning

The learning method of LVQ-NN is often called competition learning algorithm, because for each training pattern the reference vector that is closest to it is determined. The corresponding output unit is also called the winner neuron. The weights of the connections to this neuron are then adapted, the direction of the adaptation depends on whether the class of training patterns and the class assigned to the reference vector coincide or not. If they coincide, the reference vector is moved closer to the training pattern, otherwise it is moved farther away. This is achieved by the following rule:

$$\Delta w_{ij} = \begin{cases} +a\,(x_j - w_{ij}) & \text{correct class} \\ -a\,(x_j - w_{ij}) & \text{wrong class} \end{cases} \quad (1)$$

2.3 LVQ Training Algorithm

Step(1): Problem definition and parameters assignment.

Step(2):Initialize weight vectors to the first m training vectors, where m is the number of different categories.

Initialize learning rate (α).

Step(3): For each training input vector x, do steps 4 to5.

Step(4):Find the network node j whose weight vector w is nearest to the current training vector x by computing the Euclidean distance D(j) between the training vector and each weight vector wj.

Step(5):Update the weights of the j neuron as follows:

If Cai=Cci Then

Wj (New)= Wj (Old)+α [X- Wj (Old)]

(i.e. move the weight vector Wj toward the input vector X).

If Cai# Cci Then

Wj (New)= Wj (Old)-α [X- Wj (Old)]

(i.e. move Wj away from X)

Where Cai and Cci are the ith actual and calculated class number.

Step(6): Reduce learning rate α.

Step(7):Test stopping condition.

The stopping condition may be a fixed number of iterations or when error reaching a sufficiently small value [7].This means that it should go to step 3 if these requirements are not fulfilled.

2.4 Genetic Algorithm and the Proposed Method

A new learning algorithm is used for training the LVQ –NN, which is the genetic algorithm (GA). GAs start by generating a population of individuals (chromosomes) each representing a point in the search space. When an initial chromosome is created by a finite number of individuals, evolution is then applied to the population, generation after generation, to produce a new population of hopefully better individuals. This process is continued until a desired solution is obtained.

2.4.1 Calculating the Fitness Function

In GAs, the fitness value of each individual corresponds to its ability to survive, and consequently needs to be evaluated. For the problem to be solved, the differences between the desired class number and the calculated class number are taken as the measurement of fitness of the individual. More precisely, the inverse design problem is transformed into minimization of the following function:

Total Sum Square Error

$$(\text{TSSE}) = \sum_{k=1}^{p} (C_a(k) - C_c(k))^2 \quad \text{.....(2)}$$

Where:

p: number of patterns.

Cc: calculated class number.

Ca: actual class number (target).

The objective function (F) is taken to be the fitness of the individual as shown in equation (2) below.

$$F = \frac{1}{TSSE} \quad \text{..... (3)}$$

This means that the inverse design problem has been converted to the maximization of the fitness function F, which is then suited for GA implementation.

2.5 The Proposed GA for Training LVQ-NN

The proposed GA may be summarized as follows:

Step(1): Problem definition and parameters assignment.

Step(2): Create an initial population of chromosomes and set current population to this initial population. Initialize the crossover probability Pc, the mutation probability Pm, and the number ofpopulation N.

Step(3): Compute the total sum square error (TSSE) for all chromosomes

Step(4): Evaluation, where each chromosome is evaluated according to its value of the fitness function F=1/TSSE.

Step(5): Save best population (beginning of the elitism technique).

Step(6): Selection in which parents for reproduction based on their fitness are selected.

Step(7):Recombination, which generate two offspring from the two parents chromosomes through the exchange of real strings (crossover).

Step(8): Mutation, which apply a random mutation to each offspring.

Step(9): Replace old population with new population

Step (10): Copy best population to the new population in the next generation.

Step (11): Go to step (3), if some criterion is not reached.

3. Principal Component Analysis Neural Network (PCA-NN)

The principal component analysis or Korhunen-Loeve transform is a mathematical way of determining the linear transformation of a sample of points in N-dimensional space [8].

PCA is a technique to find the direction in which a cloud of data points is stretched most. These directions represent most of the information in the data and are thus important to know. Knowing these directions allows the data in a compressed form and later reconstructing the data with amount of distortion [8].

PCA is a very well known statistical procedure that has important properties. Suppose that we have input data of very large dimensionality. We would like to project this data on to a smaller-dimensionality space. Projection always

distorts our data somewhat. The linear projection that accomplishes this goal is the PCA. The PCA-NNs are feedforward unsupervised Hebbian learning networks. In unsupervised learning, there is no external teacher. A feedforward neural network can be without a teacher according to some learning rule, which imposes a certain condition on its output [9].

3.1 The GHA for the PCA-NN

The GHA may be summarized as follows:

Step(1): Problem definition and parameters assignment.

Step(2): Initialize the weights of the network wji to small random values at n=1.

Step(3): Assign a small positive value to the learning rate parameter a ($0<a<=1$).

Step(4): For n=1,i=1,2,…,M, and j=1,2,…L, compute

$$yj(n)=\sum_{i=1}^{M} w_{ji}(n)x_i(n) \quad (4)$$

$$zj(n)=\sum_{i=1}^{m} y_j(n)w_{ji}(n) \quad (5)$$

$$\overline{X}(n)=xi(n)-\sum_{k=1}^{j-1} z_k(n) \quad (6)$$

$$\Delta w_{ji}(n) = a y_j(n)\overline{x}_i(n) \quad (7)$$

$$w_{ji}(n+1)=w_{ji}(n)+\Delta w_{ji}(n) \quad (8)$$

step(5): Increment n by 1

step(6): Go to step(4), and continue until the synaptic weights wji reach their steady state values

3.2 The Proposed GA for the PCA-NN Training

The proposed GA works as follows: A population of individuals is generated by randomly selecting different genes. The fitness of each individual is then selected according to the equation below:

$$Fitness=\frac{1}{TSSE} \quad (9)$$

And the technique applied to the high fitness individuals to generate a new population. The cycle of evaluate continues until a satisfactory solution, hopefully optimal, is found.

Tournament selection is used to overcome the problem of fitness scaling with direct comparison of fitness value. This type of selection is generally considered more efficient and more robust than roulette wheel [1]. The penalty function is used in the present work, in which the individuals are penalized with a very large positive constant if the error is greater than the feasible limit (i.e., do not survive in the next generation) [10]. The proposed GA for PCA-NN (unsupervised) training may be summarized as follows:

Step(1): Problem definition and parameters assignment.

Step(2): Create an initial population. Initialize Pc, Pm

Step(3): Compute TSSE values for all chromosomes as in equation (3).

Step(4): Compute fitness values as in equation (9)

Step(5): Save best population

Step(6): Selection

Step(7): Crossover

Step(8): Mutation

Step(9): Replace old population with new population.

Step(10): Copy best population to the new population in the next generation

Step(11): If maximum number of allowed generations or the prescribed error level is reached, then the process stops else continue from step(3).

3.3 The Proposed Hybrid Neural/Genetic Algorithm

A hybrid learning algorithm is consist of two learning stages, the first learning stage uses genetic algorithm with feed forward step to accelerate the whole learning process. The genetic algorithm performs global search and seeks near optimal initial point (weight vector) for the second learning stage which uses generalized Hebbian algorithm (GHA). Here, each

chromosome is used to encode the weights of neural network.
The hybrid algorithm may be summarized as follows:
1.Run the proposed GA for PCA-NN to select near-optimal initial weights values.
2.Set the best chromosome as the initial weight vector to GHA.
3.Run the GHA.

4. Experimental Results

Study case 1.

The input signatures shown in Fig.(2) with (175×245) pixels were used to train the LVQ-NN. An LVQ-NN architecture of 8 input units and 3 output units was used to solve the signature recognition problem with three different classes (three persons).

A. Training Phase

The corresponding target vector for each input signature in Fig.(2) is shown in table (1).

Test1. LVQ-NN learned using competitive algorithm

Training parameters:
No. of Classes: 3
No. of Signatures: 5
No. of output units = No. of classes
Learning rate = 0.2

Test 2. LVQ-NN learned by GA.

Training parameters:
No. of classes (persons)= 3;
No. of signatures = 5
Population size = 20
Mutation rate = 0.01
Crossover rate = 0.6
Length of chromosomes = 8×3
Training results are shown in table (3).
Training results are shown in table (2).
Fig.(3) shows the comparison of convergence between competitive learning and the proposed GA.

B. Recognition Phase

The LVQ-NN trained by the previous step was tested by apply the images shown in Fig. (4), and gives 100% classification as shown in table (4).

Study case 2. The second example use matrix A (3×3)

$$A=\begin{bmatrix} 1.194 & 0.0018 & -0.004 \\ -0.0491 & 1.2133 & -0.0279 \\ -0.0040 & 0.001 & 1.1987 \end{bmatrix}$$

• Each column represent pattern (p=3).
• Number of input nodes to the PCA-NN equal to the number of rows (M=3)
• Number of output nodes changeable (L<3)

$Rx = A \times A^T$, the eigenvalues of Rx are ..... (10)

$$Eig(R_x)=\begin{bmatrix} 1.5181 \\ 1.4424 \\ 1.3773 \end{bmatrix} \quad .....(11)$$

CASE 1. When the input nodes (M=3), and the number of output nodes (L=2), the training parameters of the algorithms shown in table (5)

$$\text{Optimal SSE}=\sum_{i=2+1}^{3} l_i = \lambda_3 = 1.3773 \quad (12)$$

Number of generation (Ng) = 100
The results after many runs are shown in table (5), Figure (5) shows the comparison of convergence between algorithms for the case 1, run 5, and shows that the proposed hybrid method is best from other algorithms.

CASE 2. When the input nodes (M=3), and the number of output nodes (L=1), the training parameters of the algorithms shown in table (6)

$$\text{Optimal SSE}=\sum_{i=1+1}^{3} l_i = l_2 + l_3 = 2.8197$$

Number of generation (Ng) = 60
The results after many runs are shown in table (7). Figure (6) shows the comparison of convergence among GHA, GA and HNGA.

5. Conclusions

This study has focused on training the LVQ-NN and the PCA-NN by GA.
A new scheme is presented for signature recognition system using invariant moments with LVQ-NN.
The following conclusions are drawn from this work:

1-GA has been used successfully to train LVQ-NN because it reaches quickly to the region of the optimal solutions.
2-GA for training LVQ-NN is independent of the parameters that the competitive algorithm depends on. Tests show that GA obtain best weight vectors and is able to classify data from a specific situation with a small number of epochs.
3-The simplest suitable method of initializing the weight (reference) vectors greatly improves the performance of training LVQ-NN in both competitive algorithm and genetic algorithm.
4-The proposed signature system is used to identify persons. It uses a competitive neural network with invariant moments to verify the signature. The system can be used to recognize a shift, scale and rotation invariant image. This system was tested on many signatures and the following remarks can be concluded:
► The system is very sensitive to changes in the person signatures.
► The time required for training LVQ-NN is depends on the algorithm, size and number of trained images.
► Recognition of 80% result is achieved using one sample for each person, however, this percentage increases to more than 90% if more than one sample is taken for each person.
5-In the PCA-NN there is no need to compute the correlation matrix as required by the competitive learning technique. The Eigen vectors and Eigen values are derived directly from the input vector. When the number of input data is large, this advantage becomes significant because computation of correlation matrix is time consuming.
6-The GHA, GA and HNGA are used to train PCA-NN. Tests show that HNGA:
► is significantly faster than the GHA, GA algorithms.
► producing a lower TSSE in the same number of iterations.
► it also has the advantage, that it can avoid dropping in the local minima during searching process.

6. References

[1] T. L. Rollins and F. L. Wcichmai, Phys. Status Solid.. 7.5 (1996) 233.
[2] F. I. Kreeingold and B. S. Kulinkin, Son Phys. Semicond. 4 (1971) 2022.
[3] J. Tominaga; .J. Phys.: Condens. Matter 25 (2003) .
[4] U. K. Barik, S. Srinivsan; C. L. Nagendra and A. Subrahmunyam; Solid Films 429 (2003) 129.
[5] D. Hecht, P. Borthen and H. Strehblow; Sin-f. Sri. 365 (1996) 263.
[6] L.A. A. Petteresson and P. G. Snyder, Thin Solid Films 270 (1995) 69.
[7] U. K. Barik and A. Subrahmanyam, in Proc. 11th Int. Workshop Phys. Semiconductor Devices. eds. V. Kumor and P. K. Basu, 11-I5 December 2000, New Delhi, India, pp:12.
[8] E.Tselepis and E.Fortin, J.Mater. Sci:.21(1986).
[9] D. Song, ,I. Zhao, A. Wang. P. Widcnborg, AV. Chin and A. G. Abcrle, 17th Ear. PV Conf. (Munich, 2001)
[10] R. A. Krohling, H. Jaschek and J. P. Rey, "Designing PI/PID Controllers for a Motion Control System based on GAs", Proceeding of the 12th IEEE, International Symposium on Intelligent Control, Istanbul, Turkey, 1997

Table (1) Target Vector of study case1.

Input Sig.	Target
OMER(Sig1)	Class 1
BESHAR(Sig2)	Class 2
WESAM(Sig3)	Class 3
OMER(Sig4)	Class 1
WESAM(Sig5)	Class 3

Table (2) Training Results of study case1.

Epochs	2	5	7	9	11
TSSE	1	1	1	1	0

Table (3) Training Results of Test 2.

Epochs	1	2	3	4
TSSE	1	1	1	0

Table (4) Test results of study case1 (Recognition Phase).

Images	TSSE	Actual class	Calculated class	Result
Sig.1	3.18e-5	2	2	Pass(BESHAR)
Sig.2	1.4e-4	3	3	Pass (WESAM)
Sig.3	1.686e-5	3	3	Pass(WESAM)
Sig.4	1.8e-2	2	2	Pass(BESHAR)

Table (5) Training parameters of study case2(case1).

Algorithm	*Training parameters*
GHA	α =0.4
GA	N_p=80, Pc=0.8, Pm=0.05
HNGA	N_p=80, Pc=0.8, Pm=0.05, α =0.4

Table (6) Training parameters for study case2

Algorithm	*Training parameters*
GHA	α =0.4
GA	N_p=80, Pc=0.8, Pm=0.05
HNGA	N_p=80, Pc=0.8, Pm=0.05, α =0.4

Table (7) TSSE for study case2 (case1).

Run no.	GHA	GA	HNGA
1	TSSE=2.8203 $\lambda_i=[1.5175]$	TSSE=2.8202 $\lambda_i=[1.5156]$	TSSE=2.8197 $\lambda_i=[1.5181]$
2	TSSE=2.8388 $\lambda_i=[1.4993]$	TSSE=2.8203 $\lambda_i=[1.5346]$	TSSE=2.8197 $\lambda_i=[1.5181]$
3	TSSE=2.8207 $\lambda_i=[1.5172]$	TSSE=2.8228 $\lambda_i=[1.5651]$	TSSE=2.8197 $\lambda_i=[1.5181]$
4	TSSE=2.8246 $l_i=[1.5053]$	TSSE=2.8365 $\lambda_i=[1.4905]$	TSSE=2.8198 $\lambda_i=[1.5181]$
5	TSSE=2.8327 $\lambda_i=[1.5041]$	TSSE=2.8224 $\lambda_i=[1.4830]$	TSSE=2.8198 $\lambda_i=[1.5181]$

Table (8) TSSE for study case2 (case2).

Run no.	GHA	GA	HNGA
1	TSSE=1.3781 $\lambda_i=\begin{bmatrix}1.5181\\1.4416\end{bmatrix}$	TSSE=1.3834 $\lambda_i=\begin{bmatrix}1.5739\\1.3938\end{bmatrix}$	TSSE=1.3774 $\lambda_i=\begin{bmatrix}1.5181\\1.4423\end{bmatrix}$
2	TSSE=1.3791 $\lambda_i=\begin{bmatrix}1.5181\\1.4406\end{bmatrix}$	TSSE=1.3957 $\lambda_i=\begin{bmatrix}1.5405\\1.4017\end{bmatrix}$	TSSE=1.3774 $\lambda_i=\begin{bmatrix}1.5181\\1.4423\end{bmatrix}$
3	TSSE=1.378 $\lambda_i=\begin{bmatrix}1.5181\\1.4417\end{bmatrix}$	TSSE=1.3953 $\lambda_i=\begin{bmatrix}1.6572\\1.3885\end{bmatrix}$	TSSE=1.3776 $\lambda_i=\begin{bmatrix}1.5181\\1.4421\end{bmatrix}$
4	TSSE=1.3849 $\lambda_i=\begin{bmatrix}1.5181\\1.4417\end{bmatrix}$	TSSE=1.3840 $\lambda_i=\begin{bmatrix}1.6572\\1.3885\end{bmatrix}$	TSSE=1.3774 $\lambda_i=\begin{bmatrix}1.5181\\1.4421\end{bmatrix}$
5	TSSE=1.3889 $\lambda_i=\begin{bmatrix}1.5181\\1.4309\end{bmatrix}$	TSSE=1.3845 $\lambda_i=\begin{bmatrix}1.5648\\1.4736\end{bmatrix}$	TSSE=1.3773 $\lambda_i=\begin{bmatrix}1.5181\\1.4423\end{bmatrix}$

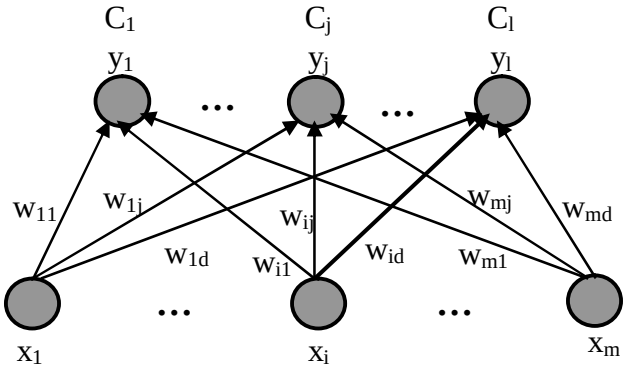
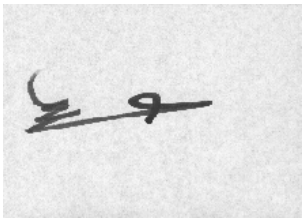
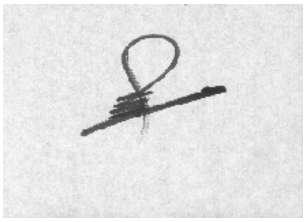


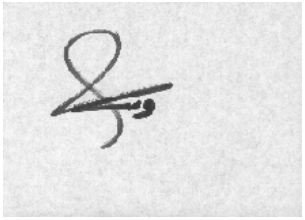
Figure (1) LVQ Neural Network



(a)OMER(Sig.1)

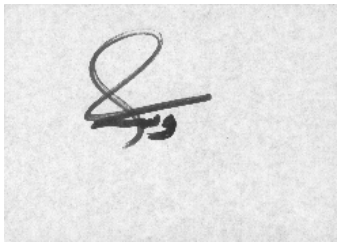


(b)BESHAR(Sig.2)



(c)WESAM(Sig.3)

(d)OMER(Sig.4)



(d)WESAM(Sig.5)

Figure (2) Signatures used in Training Phase of study case1.

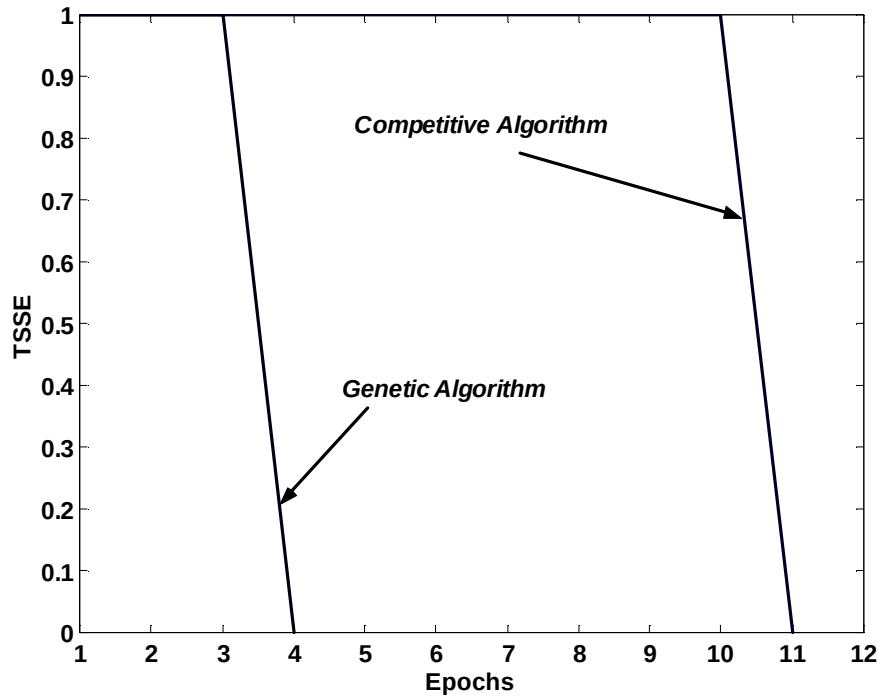


Figure (3)Performance Comparison among Competitive Algorithm and the proposed GA for study case 1

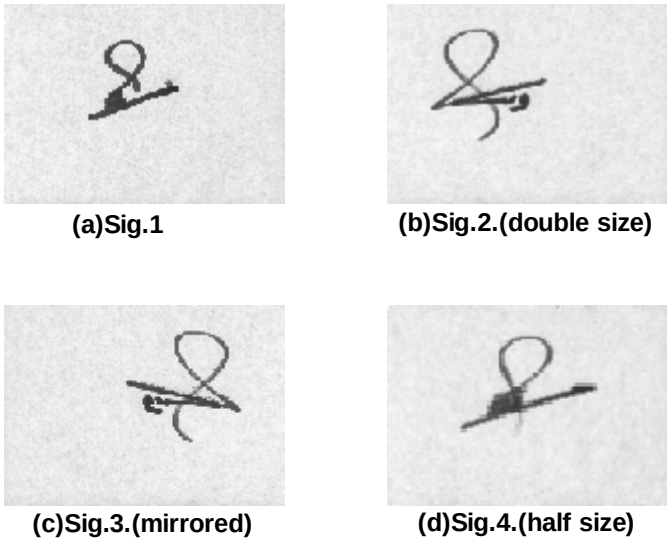


Figure (4) Signatures(Test images) used in Recognition Phase of LVQ-NN of study case 1.

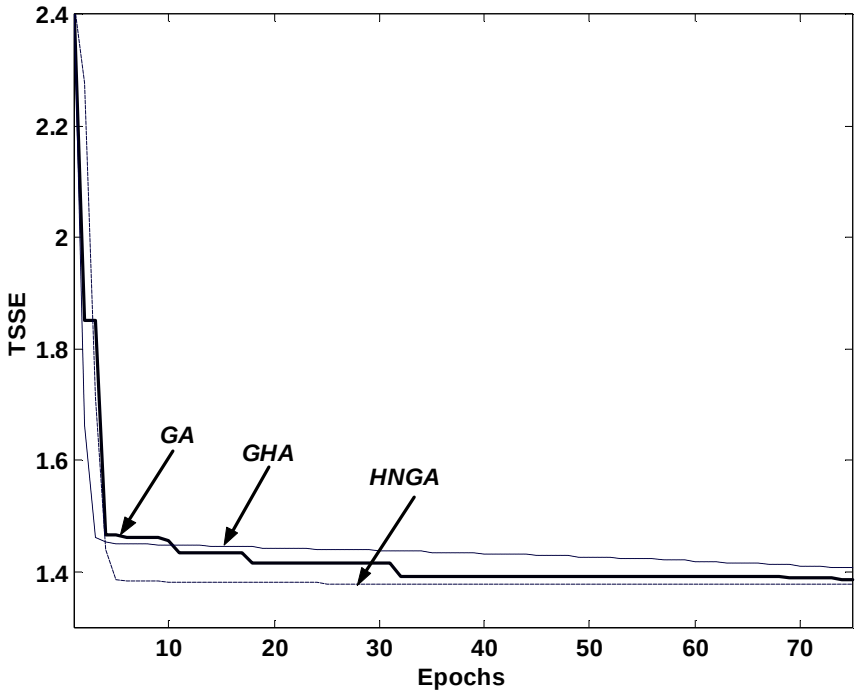


Figure (5) Performance comparison among GHA, GA and HNGA of study case2, case1, run 5.

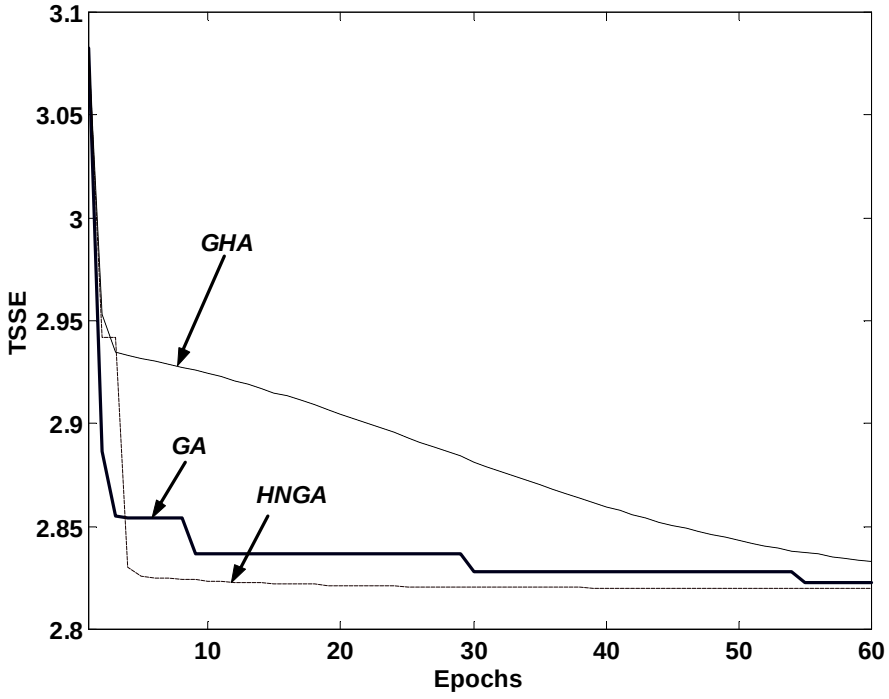


Figure (6) Performance comparison among GHA, GA and HNGA of study case2, case2, run 5.