Mansoura Engineering Journal

Volume 35 | Issue 1 Article 7

11-17-2020

Learning of Artificial Neural Networks by Genetic Algorithms.

Jamal Abdul Fatah Azzam
Department of Electrical Engineering, Faculty of Engineering, Mansoura University, Mansoura, Egypt

Learning Of Artificial Neural Networks by Genetic Algorithms

تعليم الخلايا العصبية الصناعية باستخدام خوارزمات الجينات الوراثية

Jamal Abdul Fatah M. Azzam

Key words:

Artificial Neural networks, evolutionary computation, genetic algorithms, fitness function.

ملخص:

ان التعليم و التطور هما صورتان أساسيتان للذكاء الصناعى . وفى السنوات الأخيرة هناك اهتمام كبير فى دمج التعليم و التطور مع الخلايا العصبية الصناعية.

ان تعليم الشبكات العصبية ذات التغذية الامامية هى عملية تقدير معاملات لا خطية يمكن حلها بالعديد من تقنيات الحلول المثلى . وكثير من الأبحاث فى مجال الخلايا العصبية يركز على استخدام العديد من طرق الهبوط المنحدر . ومن المعروف أن استخدام مثل هذه التقنيات يكون بطيئا ، كما أنه لاينتج عنه تحسن ملموس فى النتائج اذا استخدم فى المسا ئل المعقدة .

وفى هذا البحث نقدم خوارزماً جديداً مقترحاً لتعليم الخلايا العصبية ، وهو يعتمد على توظيف قوة تقنيات التطورالوراثى لتعديل قيم الأوزان فى الخلايا العصبية. والأمثله التى تم محاكاتها باستخدام الخوارزم المقترح تنتج حلولا مثالية أو شبه مثالية وفى وقت حساب صغير.

Abstract:

Learning and evolution are two fundamental forms of artificial intelligence. There has been a great interest in combining learning and evolution with artificial neural networks (ANNs) in recent years. The training problem for feed forward neural networks is a nonlinear parameter estimation that can be solved by a variety of optimization techniques. Many researches in the literature on neural networks has focused on variants of gradient descent. The training of neural networks using such techniques is known to be a slow process, with more sophisticated problems not always performing significantly better.

In this paper a new proposed algorithm to learn the neural networks is introduced. This algorithm implements the effectiveness of the genetic evolution techniques to adjust the weights values of the feed forward neural networks. Simulation examples of the proposed algorithm produce optimal or suboptimal solutions in a small computation times.

Accepted March 31, 2010.

1. *Introduction:*

The evolving interest in Artificial Neural Networks (ANNs) in the scientific community is fueled by many successful and promising applications. ANNs can be implemented in tasks of optimization [1], speech recognition [2], pattern recognition [3], signal processing [4], function approximation [5], control problems [6,7], financial modeling [8] etc. . Even though ANNs are capable of performing a wide variety of tasks, yet in practice sometimes they deliver only marginal performance. Inappropriate topology selection and learning algorithms are frequently blamed [9]. There is little reason to expect that one can find a uniformly best algorithm for selecting the weights in a feed forward or feedback (recurrent) ANNs. Any elevated performance over one class of problems is exactly paid for in performance over another class [10, 11, 12, 13]. To improve the performance the networks formation can be changed. Fig.1 summarizes the architecture taxonomy of ANNs.

Learning of ANNs is a crucial problem that gains a lot of researchs in the last decades. Several techniques are implemented in this regard. Each has its application areas. The weights values, the architecture of the network and the activation functions all are key factors that determine the output for input/output patterns. The general shortfalls of these techniques lies in: Long learning time (a range of hours, may be days) [25:31], with total error (up to 15%) [26]. Fig. 2 shows some of the most frequently used activation functions which represent the Soma in In ANNs each weight value affects all the subsequent outputs. Applying genetic algorithms (GAs) in minimizing the overall error in ANNs faces a great obstacle that is: how much each weight shares in the output error. This problem becomes more complex if the ANNs that have multi inputs multi outputs (MI/MO) Fig. 3.

In this paper we introduce a new algorithm based on the genetic evolution to train the feed forward networks. A fitness function explained in [14] is implemented here to determine the share value of each weight in the total error of all I/O patterns, consequently, the weights values can be adjusted gradually in the GAs principal "survival for the fittest" to their optimal values. , the genetic algorithm is implemented to optimize the weights values for MI/MO. The simulation results optimal or suboptimal weights values. The computation time is considerably small.

2. Learning Algorithms:

Ability to learn is a fundamental trait of intelligence. Although what is meant by learning is often difficult to describe, a learning process, in the ANNs can be viewed as the problem of updating network architecture and connection weights so that the network can efficiently perform a specific task [15]. The learning process must address specific issues e.g. ***Learning paradigm***: what information is available to the ANNs. ***Learning rules:*** the rules which govern the updating of the system. ***Learning algorithm***: refer to the procedure in which learning rules are used for the biological counterpart neurons. adjusting weights. ***Sample capacity*** :

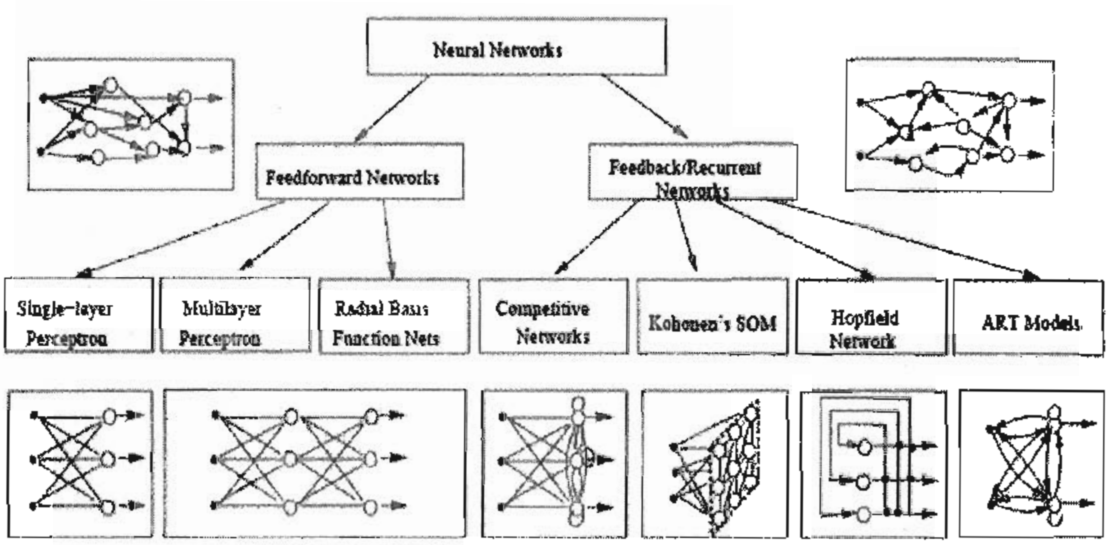

Fig. 1 Taxonomy of Network Architecture

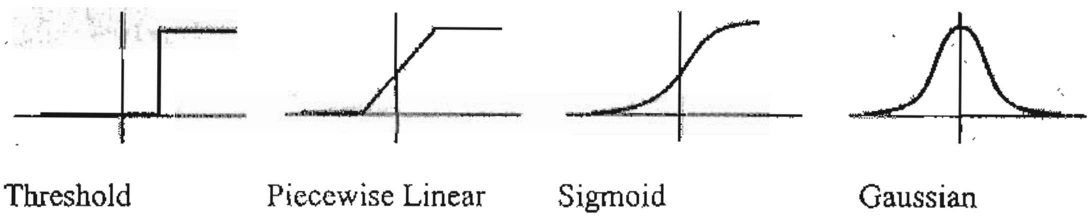

Fig.2 Different Types of Activation Functions

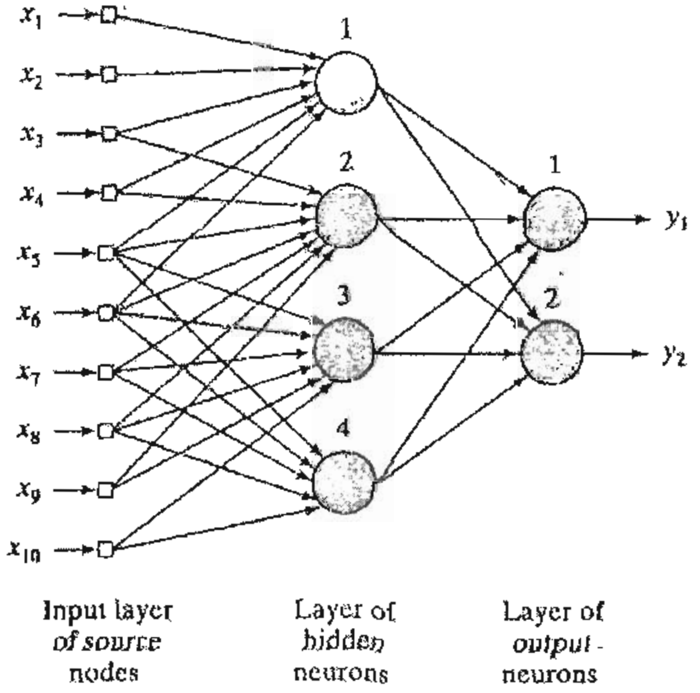

Fig.3 Multi layer ANN

how much the network can learn from examples? and ***time complexity:*** *how fast the system can learn.*

Capacity, sample complexity, time complexity forms the learning theory [16]. There are three main learning paradigms:

(i) Supervised: the network is provided with a correct answer to every input pattern. Weights are determined so that the network can produce answers as close as possible to the known correct answers. Reinforcement learning is a special case of supervised learning where the network is provided with critiques on the correctness of the output, not the correct answers themselves.

(ii) Unsupervised learning: the correct answers are not required with each input pattern in the training data set. It explores the underlying structure in the data, or correlation between patterns in the data, and organizes patterns into categories from these correlations.

(iii) Hybrid learning: combines supervised learning and unsupervised learning. Typically, a portion of weights are determined using supervised learning, while the others are obtained from unsupervised learning.

3. Genetic algorithm:

The genetic algorithm is a stochastic optimization algorithm that was originally motivated by the mechanisms of natural selection and evolutionary genetics. Over the last decade, GA has been extensively used as search and optimization tools in various problem domains, including: science, commerce and engineering. The primary reasons for ease of use and global perspective. There are some differences between the GA and traditional searching algorithms. They can be summarized as follows [17,18]:

- The algorithm works with a population of strings, searching many peaks in parallel, as opposed to a single point.
- The GA works directly with strings of characters representing the parameter sets, not the parameters themselves.
- The GA uses probabilistic rules instead of deterministic rules.
- The GA uses objective function information instead of derivatives or other auxiliary knowledge.

GA is inherently parallel, because it simultaneously evaluates many points in the parameter space (search space). So, the GA has a reduced chance of converging to local optimum and would be more likely to converge to global optimum. The GA requires only information concerning the quality of the solution produced by each parameter set (objective function values). This differs from many optimization methods which require derivative information or, worse yet, a complete knowledge of the problem structure and parameters. Since the GA does not require such problem specific information, it is more flexible than that most search methods [19]. Typically, the GA is characterized by the following components:

- A genetic representation (or an encoding) for the feasible solution to the optimization problem.
- A population of encoded solution.
- A fitness function that evaluates

their success are their broad applicability,

the optimality of each solution.

- Genetic operators that generate a new population from the existing population.
- Control parameters.

The basic flow chart of the GA is illustrated in Fig. 4 where ($\varepsilon > 0$) a small number to check convergence.

The design of ANNs using GAs can be helpful in terms of two main issues. First, it automates the design of the network which would otherwise have to be done by hand using trial and error [20]. Second, the process of the design can be analogous to a biological process in which the ANNs blueprints in chromosomes develop through an evolutionary process. Designing networks by hand may be very complex. Even through a design is found to be sufficient for a task by trial and error, the risk of missing more promising architecture is not eliminated. Given the complex combination of performance criteria, such as learning speed, compactness, generalization ability, and noise-resistance, it is very difficult to optimize a network design. The problem of designing an ANN for a specific problem involves searching the space of architectures for one which will perform best in meeting the requirements of the problem. The search space for such a problem may be infinitely large, un-differentiable, complex, noisy, deceptive and multi-modal.

GAs are applied to neural networks in two different ways: they either employ a fixed network structure (i.e. the number of nodes and the connections among them are fixed) with connection weights under itself [21:23].

4. The proposed Algorithm:

Given groups of inputs and desired outputs pairs (I_i, O_{di}); i = 1 to n as shown in Fig.5 e.g.

$(I_1, I_2, I_3, .. \quad , O_{d1}, O_{d2}, O_{d3}, ..)_{\text{pattern } j=1}$,

$(I_1, I_2, I_3, .., O_{d1}, O_{d2}, O_{d3}, ..)_{\text{pattern } j=2}$, ...

Compute the minimum number of neurons weights W_{ij} in the hidden layer and the optimal values of the weights in the hidden layer (or layers) and in the output layer such that the sum of errors of the actual outputs O_i and the desired outputs O_{di} is minimum. Referring to Fig. 5 :

i denote the row number.

j denotes the vertical number.

g number of input/output patterns.

The error function is:

$$E_{total} = \frac{1}{2}\sum_{j=1}^{g}\sum_{i=1}^{n}(y_{di} - y_i)^2 \qquad (1)$$

The sum of absolute errors can be used correctly although the form in (1) is easy and frequently used [24].

The learning algorithm is explained in the following steps:

4.1 Specify the suitable activation function to be used.

4.2 Assume the order of weights vector in the hidden layer = the maximum order of inputs vectors.

4.3 For the two parents P1, P2 randomly choose small values for all weights in the hidden layer and in the output layer $i.e. W_{ij} \approx (0.03 : 0.3)$.

evolutionary control, or they are used in designing the structure of the network

4.4 Check if $E_{total} \leq \varepsilon$ then stop.
The weights values are optimal in this case.

4.5 Using the first I/O group, compute the error $e_i = O_{di} - y_i$ randomly choose two weight values and change them to close values.

4.6 Repeat steps 4.4 and 4.5 for all (g) I/O groups. Compute E_{total} in equ. (1).

4.7 Compute the derivatives of E_{total} w.r.t. each weight W_{ij} by making a small random perturbation in each weight individually $W_{ij} = W_{ij} + \Delta W_{ij}$.

$\Delta W_{ij} \leq 0.05(\Delta W_{ij})$.

4.8 For each increment of a single weight compute:

$$\frac{\delta E_{total}}{\delta W_{ij}} = \frac{\Delta E_{total_{ij}}}{\Delta W_{ij}} \qquad (2)$$

Where: $\Delta E_{total_{ij}}$ is the difference in errors for all I/O groups due to the small perturbation in the weight number i, j.

and the sum of the derivatives of all errors w.r.t. W_{ij} is:

$$d_{total\, ij} = \sum_i \sum_{ij} \frac{\delta E_{total_{ij}}}{\delta W_{ij}} \qquad (3)$$

4.9 Compute the fitness function of each weight:

$$Fitness\_W_{ij} = \frac{d_{total\, ij}}{E_{total}} \qquad (4)$$

4.10 Compute the repetition rate of the weight value

$$RR\_W_{ij} = \frac{Fitness\_W_{ij}}{E_{total}} \qquad (5)$$

4.12 Repeat steps from 4.3 up to 4.11 till the number of iterations $\geq$ sufficient number e.g. 100 iteration. If this is true while E_{total} still $> \varepsilon$ then increase the weights in the hidden layer by 1. And go to 4.3 again.

<u>The algorithm can be summarized as:</u>

```
READ    the activation function
READ    max. number of weight per layer
INITIALIZE  randomly the values of the
            weights
REPEAT till  the minimum total error
             (of all I/O patterns) ≤ ε .
INITIALIZE number of weights = number
           Of inputs (in one I/O pattern)
DO      till number of iterations ≥ max
        number of iterations

           Total Error = outputs for all I/O
           patterns - Actual outputs for all
           I/O patterns
RUN The Genetic Algorithm to minimize
           the total error
END Do.
```

Keep the weights values of the heist repletion rates in the next two children. And perform the mutation process to yield the values of the new solution.

4.11 Check if the algorithm fell in local minima. If true, apply the cross over operation. This can be done simply by

```
    If the minimum Total error (of
    all I/O patterns) $\leq \varepsilon$ then stop
ELSE
    increment the number of weights
    in the hidden layer by one.

End.
```

Genetic Algorithm:

```
INITIALIZE randomly the values of the
        weights P1 (first parent)
INITIALIZE randomly the values of the
        weights P2 (second parent)

REPEAT: for all I/O patterns compute the
        outputs of the network.
```

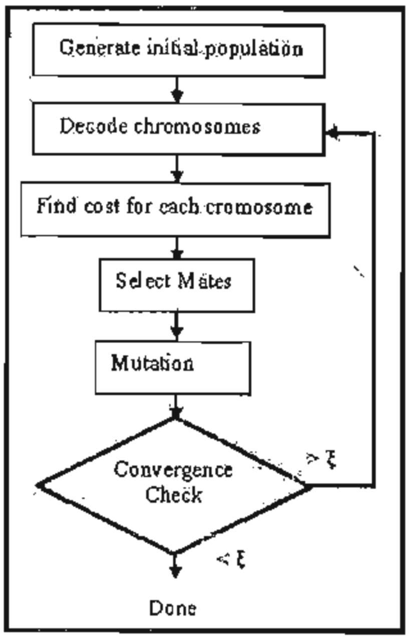

Fig. 4 The Genetic Algorithm

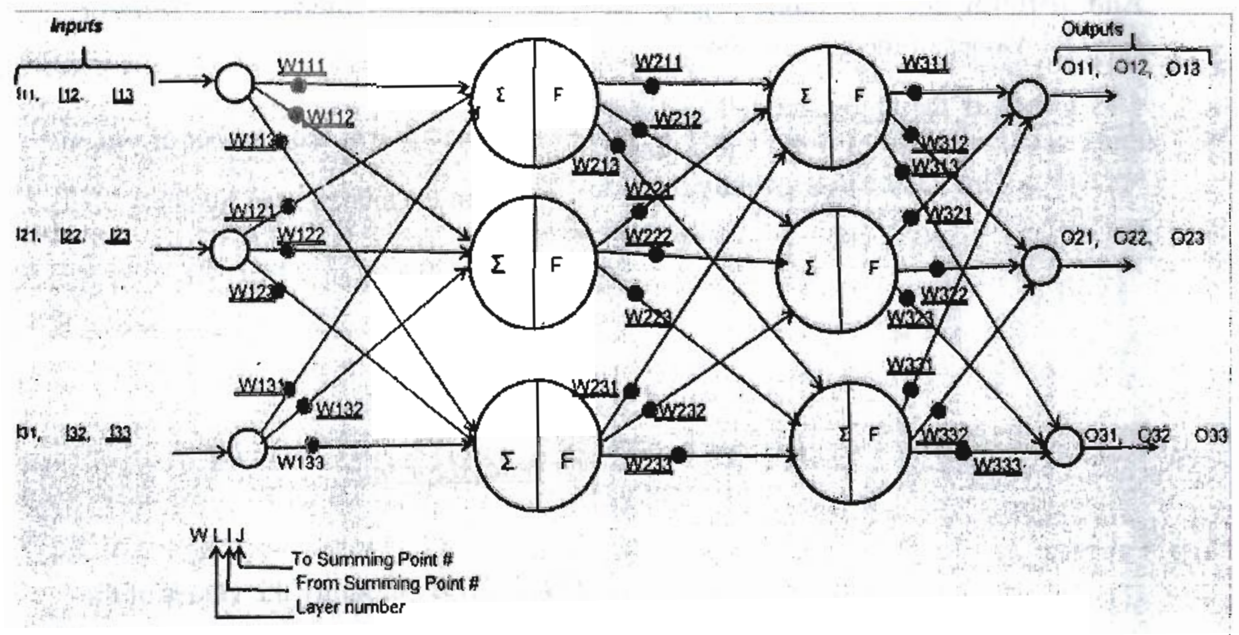

Fig. 5 Multi Pattern Configuration

```
    using weight values of P1 and their
    total error.
    the output of the network using
    weight values of P2 and their total
    Error.

END FOR

FOR P1 and P2
   compute the partial derivative of the
   total error w.r.t. each weight the Fitness
   Function of each weight the repetition
   rate of each weight

END FOR

PERFORM   Crossover

PERFORM   Mutation

COMPUTE   the new values of weights of
          P1 and P2.
```

The desired outputs are 3, 5, 7

Results:

Outputs are: 2.81236, 5.08259, 6.99616
Minimum Error is: 0.0210229
Computation Time is 275.859 Second.
The Weights values are listed in table 2.
The number of summing points is 7.

Its clear that increasing the number of a weight for one hidden layer improve the results and consumes more computation time compared to increasing the hidden layers.

0.153966	0.10155	0.0980102

```
            pick up the minimum  error
UNTIL    number of iterations > Max
            number of iterations
END REPEAT.
```

5. Simulation Results:

Example 1:

Inputs are: 2, 3, 4.
Desired Outputs 3, 4, 7.
Number of hidden layers is 2
Results:
The outputs are: 2.88517, 4.77561, and 6.71453.
Total error is 0.0725144
The weights Wij are as shown in table 1.
Computation time is: 33.42 second.
The number of summing points is 3.

Example 2:

Using one hidden layer only and increasing the number of summing functions to 7 we get for the same inputs of example 1.

0.0834681	0.191778
0.5	0.474532
0.402326	0.324808
0.192679	0.265023
0.38435	0.255257
0.344035	0.394284
0.264229	0.5
0.5	0.0108951
0.29313	0.206
0.459258	0.5
0.416501	0.5
0.217887	0.231239
0.289575	0.109851
0.5	0.5
0.5	0.319697

0.478072	0.158254	0.363399
0.233177	0.389142	0.448759
0.486587	0.383007	0.153401
0.317316	0.423612	0.091586
0.360698	0.23983	0.0676138
0.44879	0.213095	0.176778
0.304941	0.0692618	0.260765
0.299799	0.297571	0.454833

Table 1 Weights of Example 1

0.448592	0.35551
0.439436	0.297891
0.345286	0.239921
0.465682	0.5
0.117038	0.5
0.358257	0.0628986
0.5	0.499847

Example 3:

In this example 2 input / output patterns are tested.

Inputs:

First input vector is : 2, 3, 4.

Second input vector is : 5, 2, 1

Desired Outputs are: 3, 4, 7 and 8, 10, 7

0.452818	0.135945
0.31048	0.197989
0.309107	0.301569
0.15862	0.5
0.117298	0.145146
0.493622	0.272027
0.5	0.0659505
0.421293	0.0125431
0.46556	0.461943
0.086108	0.325495
0.0782189	0.425703
0.237281	0.174505
0.349223	0.333079
0.0870693	0.213004
0.361141	0.182348
0.5	0.492615
0.00158696	0.374477
0.111057	0.409833
0.5	0.376965
0.0897549	0.290506

Table 2 Weights of Example

Minimum Error is 4. Computation time is 162.5 Second

It must be pointed out here that the signum function is used in this example to fit the binary output characteristics.
In such problems the internal computation is the same as before, when the output is less than specific value it's rounded to zero.

The results are good 4 bits error out of 130 bits (5 x 26). This small amount of error can be tolerated in data recognition.

Conclusions:

The proposed evolutionary training approach is attractive because it can handle the global search problem better in

respectively.

Actual outputs are: 4.0291, 4.7769, 2.93937 And 8.46083, 10.4153, 6.27991

Total Error is 11.7141 number of Computation time is 275 seconds.

It is clear that the error is increased with the increase of I/O number of patterns

Example 4:

Two input / output patterns are:
Inputs:
First pattern is: 2,3,4
Second pattern is: 5.2.1

Desired Outputs are: 3.5.7 and 9.11.4 respectively.

Using 1 hidden layer and 14 summing points:

Actual outputs are: 5.47272, 4.92106, 5.3143 ,9.48123, 10.1723, 4.9214
Computation time is Time is 373.734 Second
Total Error is 5.36388.

The increment of number of weights in one hidden layer is capable to reduce the total error even with the increase of number of patterns.

Example 5:

The algorithm is used for data compression of the 26 English letters.
The inputs are 5 x 7 matrix array that are restricted to either 0 or 1.The output is 5 x 1 matrix array Fig. 6.
Using a 14 summing point hidden layer we get:
a vast, complex, multimodal and non

differential manner. It does not depend on gradient information of the error, and thus it is appealing when this information is unavailable or very costly to obtain.

The most significant advantages of this technique is that:

- Computation time to learn the weights is significantly reduced (in a matter of seconds) compared to the conventional techniques.
- The weights number is minimized since the algorithm starts by the minimum number (equal to number of inputs) and increment it if the check for minimum error fails
–The error range is comparatively small.

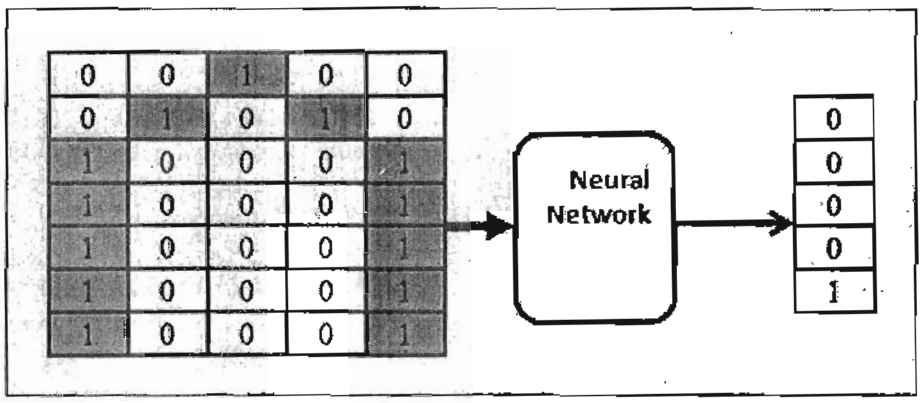

Fig. 6 Letter A compression

References:

[1] Cichocki, A. and Unbehauen, R. (1993),"Neural Networks for Optimization and Signal Processing". NY: John Wiley & Sons.

[2] Bourlard H.A and Morgan N. (1994), "Connectionist Speech Recognition: A Hybrid Approach", Boston: Kluwer Academic Publishers.

[3] Bishop C M (1995)," Neural Networks for Pattern Recognition", Oxford Press.

[4] Masters, T. (1994), "Signal and Image Processing with Neural Networks: A C++ Sourcebook, John and Sons, Inc., New York.

[5] Weigend A.S. and Gershenfeld N.A. Eds. (1994)" Time Series Prediction: Forecasting the Future and Understanding the Past", Reading, MA: Addison-Wesley.

[6] Abraham A and Nath B (1999), "Failure Prediction Of Critical Electronic Systems in Power Plants Using Artificial

Neural Networks", In Proceedings of First International Power & Energy Conference, Isreb M (Editor), ISBN 0732 620 945, Australia, December 1999.
California, Los Angeles, California, USA.

[7] Abraham and Nath B (2000), "Artificial Neural Networks for Intelligent Real Time Power Quality Monitoring Systems", In Proceedings of First International Power & Energy Conference, Isreb M (Editor), ISBN 0732 620945, Australia, December 1999.

[8] Refenes, A. (Ed.) (1995)."Neural Networks in the Capital Markets". Chichester, John Wiley and Sons, Inc., England.

[9] Ajith Abraham,(2006) "3rd International Conference on Neural, Parallel and Scientific Computations", Morehouse College, IFNA Atlanta, GA, USA.

[10] Hofgen K U (1993)," Computational limits on Training Sigmoidal Neural Networks, Information Processing Letters", Volume 46, pp. 269-274.

[11] Jones L,(1997)"The Computational Intractability of Training Sigmoidal Neural Networks",IEEE Transactions on Information Theory, Volume 43, pp 143-173.

[12] Judd S,(1990)"Neural Network Design and the Complexity of Learning", MIT Press, Cambridge, MA.

[13] Vidhyasagar M M (1997)," The Theory of Learning and Generalization", Springer-Verlag, New York.

[14] jamal A.F. azzam,(2008) "Genetic algorithm to solve multi variable functions", Mansoura Engineering Journal (MEJ), Vol. 33, No. 4, 2008.

[15] Omid M. Omidvar and et al,(1997) " Neural Systems for Control", Academic Press,University of the District of Columbia, USA.

[16] Joe Tebelskis, (1995) " Speech Recognition using Neural Networks", School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

[17] Cornelius T. Leondes, (1998), optimization Techniques", University of University of California, Los Angeles, California, USA

[18] Michael A. Arbib, (2003) " Brain Theory and Neural Networks", Massachusetts Institute of Technology, USA.

[19] D. Michie, et. al.,(1994) " Machine Learning, Neural and Statistical Classification" University of Strathclyde,-------

[20] Janusz Kacprzyk,(2008) "Studies in Computational Intelligence", Volume 83, Systems Research Institute, Polish Academy of Sciences,University of Newelska, Warsaw Poland.

[21] T. Lindblad J.M. Kinser,(2005) "Image Processing Using Pulse-Coupled Neural Networks", Springer-Verlag Berlin Heidelberg , Germany.

[22] Anil K. Jain et al.,,(1996) Artifical neural networks , IEEE computer Special Issue onNeural computing, March.

[23] Haykin S.,(1994) Neural Networks: A Comperhensive Foundation, MacMillan College Publishing Co., NY.

[24] Armingol J.M. et al,(2002) "A Genetic Algorithm For Mobil Robot Localization Using Ultrasonic Sensors", Journal of Intelligent and Robotic Systems, Vol. 34, N.2, PP 135-154.

[25] jamal A.F. azzam,(2007),"Genetic Algorithm for dynamic Task Allocation of Multi Autonomous Unmanned Air Vehicles". Mansoura Engineering Journal (MEJ), Vol. 32, No. 2, June 2007.

[26] Jose B., Cruez J.,(2003)"Genetic Algorithm for Task allocation in UAV Cooperative Control", Genshe C., AIAA

Guidance, Navigation, and Control Conference, Austin, Texas.

[27] Ibrahim Kuscu and Chris Thornton, (1994)," Design of Artificial Neural Networks Using Genetic Algorithms: review and prospect ",School of Cognitive and Computing Sciences, University of Sussex, Falmer, Brighton, UK.

[28] Belew R.K. et al.,(1992), Evolving networks using the genetic algorithms with connectionist learning, In Langton et al, editor, Artificial life II. Santa Fe Institute, CA, USA.

[29] Chalmers D.J.,(1990), "Evolution of learning: an experiment in genetic connectionism". In D. S. Touretzky, J. L. Elman, T. J. Sejnowski & G. E. Hinton (eds.), Proceedings of the Connectionist Models Summer School. San Mateo, CA: Morgan Kaufmann,USA.

[30] Hamed Hasheminia et al,(2008), "A Hybrid Methods of Neural Networks and Genetic Algorithm in Econometric Modeling and Analysis", journal of Applied Sciences, Asian network for Scientific Information, Tehran, Iran.

[31} David Gries Fred B. Schneider, (2008), "Fundamentals of the New Artificial Intelligence Neural, Evolutionary, Fuzzy and More", Second Edition, © Springer-Verlag London Limited, UK..