A New Approach to Evolutionary Optimized Fuzzy Neural Networks

MOHAMED S. IBRAHIM
Knowledge Based Systems and Robotics Department
Mubarak City for Scientific Research and Technology Applications
EGYPT

AHMED MOHAMED
Networks and Distributed Systems Department
Mubarak City for Scientific Research and Technology Applications
EGYPT

Abstract: - This paper is concerned with the evolution development of the fuzzy neural networks. The learning procedures in such networks consists of two main development phases. First, the network is evolutionary constructed through the use of evolutionary strategies and evolutionary programming. The second phase of the design of the network is aimed at its refinement and optimize the objective parameters to adapt to the nonlinear changes of the input space. There are two main optimization mechanisms working concurrently: structural development and parametric refinement mechanisms using the integrated approach based on evolutionary programming and the evolution strategies. The evolutionary-based leaning has been compared with the gradient-based learning. The proposed technique is applied for controlling nonlinear dynamics of a multilink robot system. Extensive simulations have been performed aimed at investigating the adaptive capability, tracking performance and the convergence properties of the proposed evolutionary model with respect to variety of situations. The results are remarkable compared with the fuzzy and neural networks optimized using gradient-based learning.

Key-Words: - Evolutionary optimization; Learning; Structural mutation; Tracking; Adaptability

1 Introduction
Most neural networks in literature [1,2] are trained using the standard training methods such as gradient descent techniques e.g., backpropagation. The motivations for using evolutionary computation Methods[3,4,5,6] are as follows: first, it is more suited to the training of ANNs specially for those with the applications having changing dynamical characteristics over time. For example, there is a crucial difference between classical ANN applications and ANNs for robot control [7-10]. The majority of ANN applications require one-shot mappings from input to output, that is the input vector is processed instantaneously to give an output vector. Typically, no activity is retained over time through recurrency or other mechanism. This contrasts with the task of robot control, where controllers will have to operate over time in some environment, with action at any one time typically based not only on the current sensor readings, but also sensor readings and actions in the past; ANNs for robot control are dynamical systems changing over time. It may be the case that we do not know at any one time-point what is the “correct” action to take, so gradient training methods requiring supervision are not useful. However we will still have an overall measure of how good the network has been over some extended period of time, so can use evolutionary computation methods.

Second, the evolutionary computation methods are not restricted to training the network solely in terms of the node connection weights, so
can investigate other changes such as altering the network architecture. By contrast, gradient descent methods act to minimise the current output error, so can only make changes which have a predictable impact on the output.

Third, the underlying input data received by the ANN may have a non-stationary underlying distribution. In other words the environment may change over time. Robot controllers may need to adapt to this change through some form of online “learning”. Again, little work has been done in this respect for standard training schemes, and it is not clear how such lifetime adaptation could be easily incorporated into gradient descent mechanisms. However evolutionary computation methods can be used to evolve such adaptive networks.

Finally, it is not clear what classes of ANN are suited to the generation of adaptive behaviour over time. Evolutionary computation techniques[11] can in principle work with any genotypes for which genetic operators can be defined that produce similar but not identical offspring genotypes. By contrast, as argued in the first point above, gradient descent methods act to minimise the current output error, so can only make changes which have a known impact on the output. For these reasons, applying evolutionary computation techniques to the design of artificial neural networks appears a promising avenue to explore in the design of adaptive networks.

2 Topology of the Proposed Recurrent Network

Topology of the proposed recurrent network according to the properties discussed in the introduction, we propose a network carries those features. The network is shown in Fig. 1. The signal is propagated thorough the network with the following steps:

Step 1: Each neuron generates its output as a function of the inputs of the network

$$t_i = g \left( \sum_{j} w_{ij} x_j \right)$$

Where, $t_i$ is defined as a temporal output of the (neuron=offspring)

Step 2: The output of the most fit neuron is computed as

$$y_i = g \left( \sum_{i} w_{i} x_i + \sum_{j} w_{ij} t_j \right)$$

Step 3: The output of each neuron at the output layer is computed as

$$o_i = g \left( \sum_{j} w_{ij} y_j \right)$$

3 New Idea for Explicit Representation

In ESs, the object variable vector $V_{obj}$

$$V_{obj} = (C_i , C_k)$$

Where $C_i$ is the center vector of the $i^{th}$ hidden neuron, $i \in \{1,2,...,h\}$ and $C_k$ is the centre vector of the $k^{th}$ neuron in the output layer, $k \in \{1,2,...,o\}$. The object variable vector is accompanied by a set of strategic parameters $\sigma$ where $\sigma = (\sigma_1, \sigma_2)$ It should be noted that the vector $\sigma$ is of the same dimension as $C$. Also, $\sigma_1$ has the same dimension of $C_k$. An ES-individual vector $P_l$, $l \in \{1,2,...,\mu\}$ where $\mu$ is the total number of individuals.

$$P = (C, C, \sigma, \sigma)$$

The strategic parameters are estimated explicitly from the covariance matrix $R$. 

activity to be a function of both the current input vectors and the previous states of the network.
Fig. 1 A proposed Recurrent network where the response of the ith hidden neuron is fed to itself and also the responses from others are fed to that ith neuron. The shown connections are repeated for each hidden neuron. The dotted connections means that the link of the poor fitted offsprings will be missed.

\[
R = \text{cov} X = \begin{bmatrix}
    r_{11} & r_{12} & \cdots & r_{1n} \\
    r_{21} & r_{22} & \cdots & r_{2n} \\
    \vdots & \vdots & \ddots & \vdots \\
    r_{n1} & r_{n2} & \cdots & r_{nn}
\end{bmatrix}
\]

(5)

Where \( X \) is the input vector to the input layer.

When \( \{X\} \) is an uncorrelated sequence, then:

\[ r_{ij} = \sigma_i^2 \text{ if } i=j \]
\[ r_{ij} = 0 \text{ if } i \neq j \]

(6)

In general we consider, the elements of principal diagonal of \( R \) as an estimates of the standard deviation \( \sigma_i \).

\[
\sigma = [\sigma_1, \sigma_2, \ldots, \sigma_n]^T
\]
\[
\sigma_i = \sqrt{r_i}, \quad i \in \{1,2,\ldots,n\}
\]

(7)

Take a normalized \( \sigma \) as

\[
\sigma_n = \frac{1}{N} \sqrt{r_i}
\]

(8)

Where \( N \) is the total number of samples. It should

**Remark:** The covariance matrix has an information about the mutation operator.

### 4 Integration of Evolution Strategies and Evolutionary Programming

The evolutionary process is summarised as follows:

1. **Initialization**: An appropriate size for the population is chosen. If a small population size is chosen, this will result in premature convergence. On the other hand, if a large population size is chosen this will increase the computational cost.

2. **Evaluation**: The fitness score of each individual in the population was evaluated based on the foot mean squared error (ERMS):

\[
f(p_i) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (r_i - y_i)^2}
\]

Where \( r_i \), \( y_i \) are reference signal and the actual output signal respectively, and \( N \) is the training input-output data pairs. \( f(p_i) \) is the fitness of \( i \) individual \( p \).

3. **Mutation and self adaptive mechanism**: There are two types of mutations, namely, parametric mutation and structure mutation.

Parametric mutation: The object variables and the strategy parameters undergo mutation as follows
\[ v_{w,i} = v_{w,i} + \sigma \cdot N(0, \sigma) \]  \hspace{1cm} (10)

It should be noted that, \( \sigma \) is the updated variance, it is estimated from the updated covariance matrix as discussed before. \( N(0, \sigma) \) is a zero mean Gaussian random variable with standard deviation \( \sigma \). The standard deviation variable vectors and the object variable vectors are adapted automatically and from this point of view, this mechanism is considered as self-adaptive.

Structure mutation: It is described as adding neurons and deleting the poor fitness neurons from the population pool besides maintaining the links between parents and their offsprings. Maintaining the behavioral link between parents and offsprings is an important issue.

4. Evaluation: The fitness of the offspring are evaluated as in step 2.

5. Selection: The idea is based on \((\mu + \lambda)\)-elitist strategy and ranking. We define randomly selected individuals (from the parents and offspring pool \((\mu + \lambda)\) with equal probabilities) as \( s^{(i)}, \forall i' \in \{1,...,\lambda\} \), for each \( i \). Also, let us define condition fitness that measure the competition between the two individuals as \( \beta(p, s^{(i')}) \),

\[
\beta(p, s^{(i')}) = \begin{cases} 
1 & \text{if } f(p) \geq f(s^{(i')}) \\
0 & \text{otherwise} 
\end{cases}
\]  \hspace{1cm} (11)

The total score \( sco_i \) gives as

\[ sco_i = \sum_{j=1}^{m} \beta(p, s^{(j)}) \]  \hspace{1cm} (12)

Then, the individuals are then stored according to their total score \( sco_i \) and fittest \( \mu \) solutions out of \( 2\mu \) are selected to be the parents for the next generation.

6. Checking step: The steps are repeated until we have an acceptable solution or the permissible execution time is exhausted. After the optimization process have been finished, the optimal parents \( P^* \) are obtained as,

\[ P^* = \arg \max_{i=1,...,\mu} \{ f(p_i(t)) \} \]  \hspace{1cm} (13)

5 Multilink Robot Dynamics

Multi-link robot dynamics, due to the structure of the gear box and the integrated torque sensor have a dynamical model described by the following equations [15],

\[ M(q) \dot{q} + V(q, \dot{q}) + G(q) + k(q, -q) = J'\dot{q}F \]

\[ J\ddot{q} - k(q, -q) = \tau. \]  \hspace{1cm} (14)

Where \( M(q) \) is the mass matrix, \( V(q, \dot{q}) \) is the vector corresponding to centrifugal and coriolis forces, \( G(q) \) is the gravitation vector, \( F \) is the external force applied to the end effector, \( J \) is the Jacobian matrix, \( k \) is the elasticity matrix, \( \tau \) is the motor torque vector, \( J \) is the motor and gear inertia matrix, and \( q, q_0 \) are the link and motor angular position vector. For the decoupling of the manipulator dynamics this model is transformed into new coordinate in which the joint torque is treated as a state variable instead of motor position \( k(q_0, -q_0) = \tau \). This enables separation of the dynamics dependent on the motion rate of the related state variable. Equation (14) can be re-written as:

\[ M(q) \dot{q} + N(q, \dot{q}) = \tau + J'\dot{q}F \]

\[ A\tau + B(\tau, q, \dot{q}) = \tau. \]

where,

\[ V(q, \dot{q}) + G(q) = N(q, \dot{q}) \]

\[ A = J, K \]

\[ B(\tau, q, \dot{q}) = (I + J, M')\tau + J, M' (J'\dot{q}F - N) \]  \hspace{1cm} (15)

6 Simulation Studies:

The multilink robot dynamics have been represented by single and double link robot equations to study the performance of the
The proposed evolutionary optimized neural network when working for controlling multi-link robot system.

**One link dynamics** are used as in the cited reference [12]

Convergence properties: Figs 1 and 2 show the convergence property of the one link robot using the proposed approach. In Figs. 3-8 the circles represent the values of the robot angles or angular velocities. The portraits (rays) from the origin represents the number of generations. The angles and angular velocities are represented by series1 and series2 respectively. The initial conditions in Figs. 3-8 are respectively (10,0), (10,-10), (15,5), (25,10), (-25,-10), (30,0). The target is to reach the zero circle. Figs. 9, 10 show the RMSE for self organizing algorithm based on gradient learning approach and RMSE for the proposed approach based on Evolutionary techniques respectively. The results show that the proposed approach achieves the convergence properties. Moreover, the evolutionary-based learning gives better performance index than using the gradient-based learning. Table 2 gives more comparative analysis with other models.

It shows better performance than earlier works. For double link dynamics used in the simulation, can be written explicitly as,

\[
\begin{bmatrix}
M_1 & M_1 & q_1 \\
M_1 & M_1 & q_1 \\
\end{bmatrix} + \begin{bmatrix}
-hq_1 & -h(q_1 + q_2) \\
hq_1 & 0 \\
\end{bmatrix} q_1 = \begin{bmatrix}
\tau_1 \\
\tau_1 \\
\end{bmatrix}
\]

Where

\[
\begin{align*}
M_1 &= l_1 + l_2 + m, l_1^i + m, l_2^i + 2ll, \cos q_i \\
M_2 &= l_2 + m, l_2^i + m, l_2^i + 2ll, \cos q_i \\
M_3 &= M_2 \\
M_4 &= M_1 \\
h &= m, lll, \sin q_i \\
\end{align*}
\]

The joint angle vector is \( q = [q_1, q_2] \) and the Torque vector is \( \tau = [\tau_1, \tau_2] \)

**Table 1: Dynamical parameters of a robot**

<table>
<thead>
<tr>
<th>Para-</th>
<th>l_1</th>
<th>l_2</th>
<th>m_1</th>
<th>m_2</th>
<th>I_1</th>
<th>I_2</th>
<th>F_1</th>
<th>F_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0.12</td>
<td>0.25</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Tracking performance:** Figs 11, 12 show the tracking performance of link-1 and the control signal respectively. Figs. 14, 15 depict the tracking performance of link-2 and its control signal respectively. Fig. 15 shows more tests for the tracking performance of link-1. The parameters of the proposed evolutionary network are refined and adapted to the dynamical nonlinear changes to achieve the desired tracking performance. The results prove the adaptive ability of the proposed optimized network based on the evolutionary technique. Table 3 gives comparative analysis studies between the proposed approach and earlier work. It shows superior performance over the fuzzy and neural models.

7 **Conclusion**

The proposed recurrent fuzzy neural network is optimized using the evolution strategies and the evolution programming. Both the structure and the parameters of the network are optimized using the integrated evolution technique. The developed learning algorithm estimates the mutation parameter from the actual input space. This reflects the (genetic) features of the changing environment which are encoded in the generated offspring. The proposed network has the adaptive capability and the convergence properties. The simulations results and the comparative analysis with earlier works show their superiority over the existing approaches. Also, the performance of the network using the evolutionary-based learning is better than that of using the gradient-based learning. The problems of high dimensionality must be studied in the future research using the evolutionary techniques.

---

*Proceedings of the 10th WSEAS International Conference on SYSTEMS, Vouliagmeni, Athens, Greece, July 10-12, 2006 (pp127-135)*
Fig. 1 shows the convergence property of one link robot initial conditions: (8,0), (15,10), (15,25), (25,0) for angular and angular velocities respectively.

Fig. 2 shows the performance with initial values (25,0), (-25,0), for angular and angular velocities.

Figs. 3, 4, 5 show the angular and angular velocities with initial values (10,0), (10,-10), (15,5) respectively. The objective is to reach the balance state (represented by zero circle in the figures).
Figs. 6, 7, 8 more test results with angular and angular velocities. Initial conditions (25, 10), (-25, -10), (30, 0) respectively. The objective is to reach the balance state (represented by zero circle in the figures).

Figs. 9, 10 shows the RMSE for the self organizing network using gradient-based learning, and the RMSE for the proposed network using evolutionary-based learning respectively.

Table 2: Comparative analysis of gradient based learning and evolutionary based learning

<table>
<thead>
<tr>
<th>Model</th>
<th>Variations</th>
<th>$\sigma$</th>
<th>Initial conditions $\theta$ $\theta_0$</th>
<th>Target $(0,0)$</th>
<th># Hidden neurons</th>
<th>Time(sec.)</th>
<th>Error (deg.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (25, 10)</td>
<td>(0, 0)</td>
<td>22</td>
<td>1.66</td>
<td>0.08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.5 (25, 10)</td>
<td>(0, 0)</td>
<td>15</td>
<td>1.86</td>
<td>0.04</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SONN (gradient)</td>
<td>(0, 0)</td>
<td>12</td>
<td>1.70</td>
<td>0.83</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 (25, 10)</td>
<td>(0, 0)</td>
<td>7</td>
<td>1.88</td>
<td>0.847</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 (25, 10)</td>
<td>(0, 0)</td>
<td>4</td>
<td>2.00</td>
<td>0.82</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 (25, 10)</td>
<td>(0, 0)</td>
<td>4</td>
<td>1.55</td>
<td>0.035</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Evolutionary Optimized net</td>
<td>$[\sigma_0, \sigma_0]$ estimated and optimized</td>
<td>(25, 10) more different Conditions are shown in The Figures.</td>
<td>(0, 0)</td>
<td>4</td>
<td>1.55</td>
<td>0.035</td>
<td></td>
</tr>
</tbody>
</table>
Figs. 11-15 Show the adaptive ability of the 2-link robot for the changing nonlinear dynamics.
Table 3  Case study : 2-link robotic arm ,comparative analysis with other models

<table>
<thead>
<tr>
<th>Model</th>
<th>Learning algorithm</th>
<th>Structure</th>
<th>Iterations (or generations) to converge</th>
<th>Steady state error (Ess)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feedforward NN [13]</td>
<td>BP+ Reinforcement</td>
<td>5-20-2</td>
<td>100</td>
<td>a_</td>
</tr>
<tr>
<td>NN.Robustcontrol[14]</td>
<td>BP</td>
<td>4-6-2</td>
<td>50 000</td>
<td>a_</td>
</tr>
<tr>
<td>Model[15]</td>
<td>Gradient based algorithms</td>
<td>8-56-2</td>
<td>2.5 sec</td>
<td>0.4 ° ±0.0069 rad</td>
</tr>
<tr>
<td>model[16]</td>
<td>BP+models switching</td>
<td>6-8 NN models+4</td>
<td>2 sec</td>
<td>0.12 ° ±0.002 rad</td>
</tr>
<tr>
<td>Proposed model</td>
<td>Evolution</td>
<td>4-6-2</td>
<td>1.55</td>
<td>0.034 deg.</td>
</tr>
</tbody>
</table>

NN: neural network, BP: backpropagation, a_: value not reported.

References