A Chaotic Dynamic Local Search Method for Learning Multiple-Valued Logic Networks

Gao Shangce, Faculty of Engineering, Toyama University, Toyama-shi, 930-8555 Japan
Zhang Jianchen, Faculty of Engineering, Toyama University, Toyama-shi, 930-8555 Japan
Tang Zheng, Faculty of Engineering, Toyama University, Toyama-shi, 930-8555 Japan
Cao Qiping, Tateyama Institute of System, Toyama-shi, 930-0016 Japan

Abstract: As a novel optimization technique, chaos has gained much attention and some applications during the past decade. For a given energy or cost function, by following chaotic ergodic orbits, a chaotic dynamic system may eventually reach the global optimum or its good approximation with high probability. To enhance the performance of the local search method (LS), which is based on the generalized reduced gradient algorithm, hybrid local search method is proposed by incorporating chaos. Thus, LS and chaos are hybridized to form a chaotic dynamic local search method (CDLS), which reasonably combines the searching ability of LS and chaotic searching behavior. In this paper, a CDLS method based on the logistic equation is presented to learn Multiple-Valued Logic (MVL) Networks. Simulation results and comparisons with the traditional back propagation algorithm (BP) and the standard LS method show that the CDLS can effectively enhance the searching efficiency and greatly improve the searching quality within reasonable number of iterations.

Keywords: Multiple-Valued Logic, Local search, Chaotic dynamic, Ergodic orbits, Local minimum, Global minimum, Iteration

1. INTRODUCTION

Multiple-Valued Logic (MVL) has been the subject of much research over many years [1-3]. Multiple-valued logic circuits and system, which contains multiple-valued logic circuits, multiple-valued numeric logic configuration and logic design, together with multiple-valued logic switch theory, are the main domain of the MVL [4-6]. There are kinds of methods to design Multiple-Valued numeric system, such as algebraic method [7-11], circuit method [12], theorem-proving techniques [13,14], modular design approach [15], and hyperplanes method [16], etc.

Recently, the ability of MVL networks to accumulate knowledge about objects and processes using learning algorithms makes their application in pattern recognition very promising and attractive [11, 17-20]. In particular, different kinds of neural networks are successfully used for solving the image

* Corresponding author; Contact email: ztang@eng.u-toyama.ac.jp
recognition problem [21]. Neural networks based on multi-valued neurons have been introduced in [22] and further developed in [23-26]. Multi-valued neural element (MVN) is based on the ideas of multiple-valued threshold logic [27]. Its main properties are ability to implement arbitrary mapping between inputs and outputs described by partially defined multiple-valued function. They have been proposed for solving the image recognition problems. Different models of associative memory have been considered in [23, 24, 27-29].

Although several error back-propagation based algorithms [30-33] have been proposed to emulate any arbitrary function, such as MVL functions. However, many nodes and therefore many parameters (weights and thresholds) are usually necessary to approximate an MVL function. Furthermore, these techniques cannot use any knowledge which is available prior to training. In [34,35], the authors proposed a learning MVL network that uses the prior knowledge we have on MVL network while constructing an MVL network and conducts learning in a manner analogous to neural back-propagation. In these techniques, derivatives of the node functions are required, but they generally do not exist. Furthermore, since derivatives of these piece-wise functions are zero for most inputs, learning cannot be performed efficiently. The authors have therefore, been directed towards a learning MVL network which performs learning by a non-back-propagation manner and proposed a new learning method for Multiple-Value Logic (MVL) networks using the local search method. It is a "non-back-propagation" learning method which constructs a layered MVL network based on canonical realization of MVL functions, defines an error measure between the actual output value and teacher's value and updates a randomly selected parameter of the MVL network if and only if the updating results in a decrease of the error measure [18].

However, due to their inherent local minimum problems, all these learning algorithms, either the error back-propagation based algorithms or the non error back-propagation based algorithms often converged to a local minimum solution that is far from the optimal solutions. In order to avoid these disadvantages, a hybrid local search method is proposed by incorporating chaos. Thus, LS and chaos are hybridized to form a chaotic dynamic local search method (CDLS), which reasonably combines the searching ability of LS and chaotic searching behavior. In this paper, a CDLS method based on the logistic equation is presented to learn Multiple-Valued Logic (MVL) Networks. Simulation results and comparisons with the traditional back propagation algorithm (BP) and the standard LS method show that the CDLS can effectively enhance the searching efficiency and greatly improve the searching quality within reasonable number of iterations.

The remainder of this paper is organized as follows: in the next section, the multiple-valued logic network is briefly described. Section 3 mentions the chaotic dynamic local search method; Section 4 presents the detailed process of using the CDLS method for learning MVL networks. In section 5, three typical cases using CDLS method are showed and also give the corresponding comparison results with the traditional BP and LS methods. Finally we give some general remarks to conclude this paper.
2. MULTIPLE-VALUED LOGIC (MVL) NETWORK

The value of the signal used of radix R are most commonly the extension of the positive integer binary notation, given the set Q={0, 1, ..., R-1} for any R-valued system, we find the multiple-variable MAX and MIN operators, together with many variants of single-variable literal operators widely employed.

1) MAX and MIN operators:
\[ x_1 + x_2 + ... + x_n = \text{MAX}(x_1, x_2, ..., x_n) = \text{the largest value of } (x_1, x_2, ..., x_n) \]  \hspace{1cm} (1)
\[ x_1 \cdot x_2 \cdot ... \cdot x_n = \text{MIN}(x_1, x_2, ..., x_n) = \text{the smallest value of } (x_1, x_2, ..., x_n) \]  \hspace{1cm} (2)

2) Literal operators:
\[ x(a, b) = \begin{cases} 
R - 1 & a \leq x \leq b \\
0 & \text{otherwise} 
\end{cases} \]  \hspace{1cm} (3)

The operators mentioned above together with a constant operator, for example, \( x(a, b) \), structure the algebra for functional completeness of R radix. In this condition, any R-valued function can be synthesized in a sum-of-products form.

\[ F(x_1, x_2, ..., x_n) = \sum_{e_1, e_2, ..., e_n} F(e_1, e_2, ..., e_n)x_1(e_1, e_1)x_2(e_2, e_2)...x_n(e_n, e_n) \]  \hspace{1cm} (4)

where \( x_1, x_2, ..., x_n \) are R-valued variables, \( e_i \in \{0, 1, 2, ..., R-1\}, \ i = 1, 2, ..., n \), \( F(e_1, e_2, ..., e_n) \in \{0, 1, 2, ..., R-1\} \).

We consider an R-valued logic system of \( n \) inputs \( x_1, x_2, ..., x_n \), and one output \( F(x_1, x_2, ..., x_n) \). Fig.1 shows the general realization topology for the R-valued combinatorial function, using MAX, MIN and literal operators. Node functions in the same layer are of the same type, as described below:

Layer 1: Each node in this layer is a literal function and its node function is given by Eq.4. The window parameters of the i-th input to the j-th MIN gate are defined as \( a_{ij}, b_{ij} (\ a_{ij}, b_{ij} \in \{0, 1, 2, ..., R-1\} \ \text{and} \ a_{ij} \leq b_{ij} \ (i=1,2,...,n \ \text{and} \ j=1,2,...,R^n-1)) \). The literal function for the node function is shown is Fig.2. As the values of the \( a_{ij} \) and \( b_{ij} \) change, the literal function varies accordingly, thus exhibiting various forms of literal functions.

Layer 2: A node in layer 2 corresponds to the MIN operation. Each node selects a particular area of the function and defined its function value 1 or 2 or ... or (R-1) by means of the logic signal 1 or 2 or ... or (R-1) included within the MIN term. Thus, it corresponds to m MIN nodes, maximally \( R^n - 1 \) MIN nodes. The function is given by

\[ \text{MIN}_j = \text{MIN}(c_j, x(a_{1j}, b_{1j}),...,x(a_{ij}, b_{ij}),...,x(a_{nj}, b_{nj})) \]  \hspace{1cm} (5)

where \( c_j \) is biasing parameter of \( \text{MIN}_j \) with the logic signal 1 or 2 ... or (R-1).
Layer 3: This node gives a MAX operator between the product terms:

\[ O = \text{MAX} (\text{MIN}_1, \text{MIN}_2, \ldots, \text{MIN}_m) \]  

The learning MVL network described above is a multi-layered feed-forward network in which each node performs a particular function (a node function) on incoming signals using a set of parameters specific to this node. The form of the node function varies from layer to layer, and each node function can be defined by prior knowledge on the network. Unlike the traditional neural networks, the MVL networks gives the maximal numbers of the nodes needed for any MVL functions.

3. TRADITIONAL LOCAL SEARCH (LS) METHOD FOR MVL NETWORKS

As mentioned above, any multiple-valued logic function can be synthesized in a network as shown in Fig.1 with appropriate window parameters and biasing parameters. Thus, during the learning process, these parameters can be adjusted in order to optimize an error function. For simplicity, we consider the
one output MVL problem. The error function is given by Eq.7 where \( O_p \) and \( T_p \) represent the \( p-th \) actual output value and teacher’s value corresponding to the \( p-th \) input pattern \((x_1, x_2, ..., x_n)_p\), respectively.

\[
E = \sum_p (O_p - T_p)^2
\]  

(7)

where \( p \) is the number of the total input patterns. The variable space of the MVL network consists of the window parameters \( a, b \) and the biasing parameters \( c \) \((a, b, c \in 0, 1, 2, ..., R - 1)\). Thus, if we define a vector \( V \) whose elements include all these parameters:

\[
V = \{a_{i1}, a_{i2}, ..., a_{ij}, ..., b_{i1}, b_{i2}, ..., b_{ij}, ..., c_1, c_2, ...\}^T
\]

(8)

the error function \( E \) can be expressed as:

\[
E = E(V)
\]

(9)

Then, we can iteratively adjust \( V \) to minimize the error function \( E(V) \). First, the search starts at an initial point and moves along one of \( d \) directions. \( d \) denotes the number of elements of the vector \( V \). Then the 1-th direction \( e^j \) can be defined as:

\[
e^j = (0, ..., 0, 1, 0, ..., 0)^T
\]

(10)

The sequence of iterations \( V_0, V_1, ... \) can be described as follows. For the \( k \)-th iteration, \((k \geq 0\) when \( k = 0 \), the initial iterate \( V_0 \) and the step size \( \Delta_0 \) are given\) a positive change \( \Delta V_k^+ \) in a direction \( e_k^j \) with a positive step \( \Delta_k \)

\[
\Delta V_k^+ = \Delta_k e_k^j
\]

(11)

when \( V_k^j \neq R - 1 \), \( (V_k^j \) is the j-th element in \( V \), results in a change \( \Delta E^+ \) in \( E \) as:

\[
\Delta E^+ = E(V_k + \Delta_k e_k^j) - E(V_k)
\]

(12)

Similarly, a negative change \( \Delta V_k^- \) in a direction \( e_k^j \) with a positive step \( \Delta_k \)

\[
\Delta V_k^- = -\Delta_k e_k^j
\]

(13)

when \( V_k^j \neq R - 1 \), results in a change \( \Delta E^- \) in \( E \) as:
\[ \Delta E^- = E(V_k - \Delta_k e_k^i) - E(V_k) \]  
where \( \Delta_k \) is a positive constant and usually \( \Delta_k = 1 \) for all \( k \).

Then the following learning rule:

\[
V_{k+1} = \begin{cases} 
V_k + \Delta_k e_k^i & \text{if } \Delta E^+ < 0 \text{ and } \Delta E^+ < \Delta E^- \\
V_k - \Delta_k e_k^i & \text{if } \Delta E^- < 0 \text{ and } \Delta E^- < \Delta E^+ \\
V_k & \text{otherwise}
\end{cases}
\]

will lead the network to local minimum of \( E \), and hence, to a solution of the MVL problem. Namely the error function \( E \) is always decreased by any parameter change produced in the method.

4. CHAOTIC DYNAMIC LOCAL SEARCH (CDLS) METHOD FOR MVL NETWORKS

Both the back-propagation-based learning algorithm (BP) [17] and the local search-based learning algorithm [18] have gained much attention and widespread applications in different fields. However, the performance of the BP algorithm and LS algorithm greatly depend on their initial values, and they often suffer the problem of being trapped in local optima so as to prematurely converge. In order to avoid these disadvantages, we propose a hybrid local search method of solving the local minimum problem by incorporating chaos and applied to MVL network learning.

As a kind of characteristic of nonlinear systems, chaos is a bounded unstable dynamic behavior that exhibits sensitive dependence on initial conditions and infinite unstable periodic motions. Although it appears to be stochastic, it occurs in a deterministic nonlinear system under deterministic conditions. In recent years, growing interests from physics, chemistry, biology and engineering have stimulated the studies of chaos for control [36-38], synchronizazation [39] and optimization [40-44].

One of the famous chaos system, Logistic equation, is introduced in the process of chaotic dynamic local search method defined by the following equation:

\[ cx_i(k + 1) = 4cx_i(k)(1 - cx_i(k)) \]  
where \( cx_i(k) \) denotes the \( i-th \) chaotic variable and \( k \) represents the iteration number. Obviously, \( cx_i(k) \) is distributed in the interval \((0,1)\) under the conditions that the initial \( cx_i(0) \in (0,1) \) and that \( cx_i(0) \not\in \{0.25,0.5,0.75\} \). Fig.3 shows its chaotic dynamics, where \( x_i(0) = 0.1, k = 100 \).

In this paper, the process using CDLS method was developed to get efficiently a high quality solution for learning MVL networks. And the learning rule Eq.(15) is modified as follows:

\[
\begin{align*}
V_{k+1} &= V_k + g(p_1 \cdot cV(k+1) - p_2) \\
cV(k+1) &= 4 \cdot cV(k) \cdot (1 - cV(k))
\end{align*}
\]

where \( k \) denotes the iteration number, and the function \( g(x) \) removes the fractional part of \( x \) and returns the resulting integer value. Furthermore, if \( x \) is negative, \( g(x) \) returns the first negative integer less than or equal to \( x \).
Generally, the search procedures of the CDLS method for learning MVL networks are described as follows:

Step 1. Assign MVL network;
Layer the network into literal, MIN and MAX, a three layered network as in Fig. 1.

Step 2. Initialize all the parameters;
Set all window and biasing parameters to random values in 0, 1, 2, ..., R-1; Set the maximum iteration times (Max_Epoch) and the number of the second layer (MAX_MIN).

Step 3. Present input and desired outputs;
Present all possible multiple-valued input patterns \((x_1, x_2, ..., x_n)_p\) and specify their corresponding desired outputs \(T_p\), where \(p=1, 2, ..., P\).

Step 4. Calculate actual outputs;
Use the multiple-valued operators and formulas to calculate every actual output \((O_1, O_2, ..., O_p)\) corresponding to every input vector \((x_1, x_2, ..., x_n)_p\), where \(p=1, 2, ..., P\).

Step 5. Use the CDLS learning rule Eq.(17) to adapt the windows and biasing parameters;
If the new solution is better than \(V_k = \{a_{11}, a_{12}, ..., b_1, b_2, ..., c_1, c_2, ...\}_k^T\) or the predefined maximum iteration \(Max_k\) is reached, output the new solution as the result of the CDLS and go to Step 6. otherwise, let \(k = k + 1\) and repeat Step 5.

Step 7. Use the traditional LS learning rule Eq.(15) to update the parameters.
Step 8. Repeat by going to step 3, until the window and biasing parameters stabilize.
5. SIMULATION RESULTS

In this section, we present simulation results from the application of the proposed learning algorithm and traditional algorithms to a number of multiple-valued logic functions.

The first example is a two-variables \((x_1, x_2)\) 4-valued function shown in Table 1. A canonical realization of the function can be the summation of the 11 terms:

\[
F(x_1, x_2) = 1 \cdot x_1(0,0) \cdot x_2(0,0) + 1 \cdot x_1(0,0) \cdot x_2(1,1) + 1 \cdot x_1(1,1) \cdot x_2(1,1) \\
+ 1 \cdot x_1(3,3) \cdot x_2(0,0) + 1 \cdot x_1(3,3) \cdot x_2(1,1) + 1 \cdot x_1(3,3) \cdot x_2(2,2) \\
+ 1 \cdot x_1(1,1) \cdot x_2(3,3) + 2 \cdot x_1(3,3) \cdot x_2(3,3) + 3 \cdot x_1(0,0) \cdot x_2(2,2) \\
+ 3 \cdot x_1(1,1) \cdot x_2(2,2) + 3 \cdot x_1(2,2) \cdot x_2(2,2)
\]

By using the proposed method, the parameters were set as follows:

Max_Epoch = 1000, MAX_MIN = 20, Max_k = 100, \(p_1 = 6, p_2 = 2.9\). and simulations were implemented in Visual Basic 6.0 on a Pentium4 2.8G (1G)). The learning curve is shown in Fig.4. As can be seen, the MVL network could learn the MVL function successfully. After learning, the network was adapted to a small network (Fig.6) from the initialized network (Fig.05). That is to say that the function (Eq.18) can be simplified to

\[
F(x_1, x_2) = 1 \cdot x_2(3,3) + 3 \cdot x_1(2,2) \cdot x_2(0,2) + 2 \cdot x_1(3,3) \cdot x_2(3,3) \\
+ 1 \cdot x_1(0,2) \cdot x_2(0,0) + 1 \cdot x_1(1,3) \cdot x_2(1,1)
\]

There was a reduction of 40 literal gates to 9, and 20 MIN gates to 5.

<table>
<thead>
<tr>
<th>(x_1)</th>
<th>(x_2)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Example of a quaternary function

Fig.4 Learning curve of 2-variable 4-valued function
Fig. 5 The MVL network of Table 1 before learning

Fig. 6 The final MVL network after learning

Fig. 7 The comparison of convergence performance during the proposed method, the original local search method and the neural network method on 2-variables 4-valued problem
To compare to the performance of the chaotic dynamic local search (CDLS) algorithm with other well-known back propagation (BP) and local search (LS) algorithms, we used them to learn the same MVL functions. The back propagation was performed on a three layered feedforward neural network. The input size, hidden layer size and output layer size were set to 2, 20, and 1, respectively. The network outputs with 0, 0.33, 0.66 and 1 were considered as 0, 1, 2 and 3 for 4-valued function. The weights and thresholds were initialized randomly from -1 to 1 and sigmoid function was used as the neuron's transfer function. The CDLS and LS used the completely same networks and initial conditions. The maximum iterations were 1000. Fig.7 shows the typical convergence performance of the algorithms.

For the BP algorithm, the network gave a large initial error, decreased as the learning was processed and finally converged to a local minimum. The large initial error is due to the BP algorithm's inherent disadvantages of completely randomly generated network and its initial parameters in which no knowledge on MVL function is included. Comparing to the BP algorithm, both CDLS and LS algorithm produced a relatively small error initially. This is because some knowledge on MVL functions, at least the network, node function and ranges of the initial parameters can be incorporated into the construction of initial MVL network. For the LS algorithm, we can find its error function decreased continuously, and eventually got stuck in a local minimum. Different from the tradition LS algorithm, by following chaotic ergodic orbits, the CDLS algorithm has powerful ability to escape from local minima and possible convergence to global minimum or a better solution.

Furthermore, we performed 100 simulations using the three algorithms and compared their performance in Table 2, where "Av." denotes "Average" and "Min." denotes "Minimum". And the average convergence performance is shown in Fig.8.

<table>
<thead>
<tr>
<th></th>
<th>Av. Initial Error</th>
<th>Av. Final Error</th>
<th>Min. Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP</td>
<td>80</td>
<td>10.91</td>
<td>10.62</td>
</tr>
<tr>
<td>LS</td>
<td>28</td>
<td>4.74</td>
<td>1</td>
</tr>
<tr>
<td>CDLS</td>
<td>28</td>
<td>3.40</td>
<td>0</td>
</tr>
</tbody>
</table>
We have also simulated some different classes of functions, such as 2-variable 16-valued functions and 4-variable 4-valued functions and compared them with the back-propagation networks and local search method. Fig.9 and Fig.10 show some typical simulation results of these problems respectively. As can be seen, the CDLS method can reach the optimum solution more quickly than the traditional methods. That is to say, the CDLS method has a better convergence property. Furthermore, Table.3 and Table.4 show the 100 simulations results for 2-variable 16-valued functions and 4-variable 4-valued functions respectively. All the results indicate that our algorithm performed better than the back-propagation algorithm and the original local search method on most of problems.

Table.3 Results of 100 simulations for 2-variables 16-valued problem

<table>
<thead>
<tr>
<th></th>
<th>Av. Initial Error</th>
<th>Av. Final Error</th>
<th>Min. Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP</td>
<td>24020.40</td>
<td>856.77</td>
<td>602.76</td>
</tr>
<tr>
<td>LS</td>
<td>21704.74</td>
<td>600.98</td>
<td>417</td>
</tr>
<tr>
<td>CDLS</td>
<td>21704.74</td>
<td>593.54</td>
<td>355</td>
</tr>
</tbody>
</table>

Table.4 Results of 100 simulations for 4-variables 4-valued problem

<table>
<thead>
<tr>
<th></th>
<th>Av. Initial Error</th>
<th>Av. Final Error</th>
<th>Min. Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP</td>
<td>9622.9</td>
<td>291.62</td>
<td>257.36</td>
</tr>
<tr>
<td>LS</td>
<td>267.24</td>
<td>46.32</td>
<td>38</td>
</tr>
<tr>
<td>CDLS</td>
<td>267.24</td>
<td>45.76</td>
<td>30</td>
</tr>
</tbody>
</table>

Fig.9 The comparison of convergence performance during the proposed method, the original local search method and the neural network method on 2-variables 16-valued problem
Fig.10 The comparison of convergence performance during the proposed method, the original local search method and the neural network method on 4-variables 4-valued problem

6. CONCLUSIONS

We proposed a local search method with stochastic dynamics for MVL. The proposed method can produce a simplification procession of a multiple-valued function. Furthermore, due to the introduction of chaotic dynamics, the proposed algorithm has ability to escape from local minima and eventually reaches the global minimum state or its approximation with very high probability. The learning capability of the MVL network was presented and confirmed by simulations on a large number of 2-variable 4-valued problems, 4-variable 4-valued problems and 2-variable 16-valued problems. The simulation results also showed that the proposed method performed satisfactorily and exhibited good properties and had superior ability for MVL within reasonable number of iterations.

REFERENCES

Logic, Sacramento, California, pp. 202-207.


