Identification of Chaotic Systems by Neural Network with Sandwich-like Learning Algorithm

Shing-Tai Pan\textsuperscript{1}, Ching-Fa Chen\textsuperscript{2}
\textsuperscript{1}Department of Computer Science and Information Engineering
Shu-Te University
Taiwan, R.O.C.
stpan@mail.stu.edu.tw
\textsuperscript{2}Department of Electronic Engineering,
Kao Yuan Institute of Technology
Taiwan, R.O.C.
cfchen@cc.kyit.edu.tw

Abstract—In this paper, based on genetic algorithm (GA) and steepest descent method (SDM), we proposed a sandwich-like algorithm for the learning of neural network to identify some chaotic systems. The chaotic systems interested in this paper are the duffing equation. Different identification schemes of neural network are used to identify the duffing equation. Simulation results show that performance of the algorithm proposed in this paper is more efficient than those of other methods.

I. Introduction
The chaotic system is interested to many researchers in recent years. It is discovered by a meteorologist in 1960 and is complex and unpredictable phenomena which will occur in the systems that are sensitive to their initial conditions. It is possible to get completely random results from normal equations. The chaotic system is not easy to be identified because of the character of sensitivity to the initial conditions [1]. Fortunately, artificial neural networks have self-learning capability and can identify highly nonlinear system by adjusting weights and bias between neurons [2].

Neural network (NN) is composed of highly interconnected artificial neurons and they can be used for imitation of bio-neuron system [3, 4]. It possesses learning, analogism, pattern recognition and fault tolerance capabilities, etc. The most popular architecture of neural network is back-propagation neural network (BPN) which was proposed in the 1980s. The training of BPN by steepest descent method (SDM) is to follow negative gradient direction of cost function to find out the optimal weighting and bias. However, the SDM always make the search of weights and bias fall into local optimum. Thus, we use Genetic algorithms (GA) to improve the learning performance of BPN. Genetic algorithms were developed by John Holland [5, 6] in 1970’s. It is proposed based on the Darwinian-type survival of the fittest strategy. Each individual in the population represent a potential solution to the problem at hand. They compete and mate with each other in order to produce stronger individuals. Many recent researches have point out that GA are efficient in finding global optimums of optimization problems [7]. Consequently, it is worth to investigate that how to use the GA to improve the learning performance of a neural network on the identification problem.

There have been many researches about chaos [2, 8, 9]. The neural networks trained by GA or SDM are used on prediction of nonlinear chaotic system. In this paper, combining GA and SDM, we will present a new algorithm to achieve well performance of the identification of nonlinear chaotic system.

II. The chaotic system
In this section, the dynamic of chaotic systems are introduced. It is well known that the phase space diagram and bifurcation diagram are always used to show the feature of a chaotic system [1]. Consequently, we will discuss the chaotic systems, the Duffing’s Equation, by phase space diagram and bifurcation diagram in the following subsections.

Duffing equation:

The duffing equation is a nonlinear differential equation described as
\[
\ddot{x}(t) + \alpha \dot{x}(t) + \beta x(t) + \gamma x^3(t) = u(t).
\]  \hspace{1cm} (1)

An example that can be modeled as a duffing equation is a double well oscillator which is a magneto-elastic mechanical system show in Fig. 1. This system consists of a beam positioned vertically between two magnets, with the top end fixed and the bottom end free to swing.
The dynamic of a beam with single-mode planar vibrations can be governed by equation (1), in which $u(t)$ represents the periodic forcing of the system, $x(t)$ represents the position of the flexible beam and $\alpha, \beta, \gamma$ represent the system parameters. In the following, the system (1) is transferred to a state-space model. Let $x_1(t) = x(t), x_2(t) = \dot{x}_1(t), u(t) = k \cos(\omega t)$, equation (1) can be rewritten as

$$\dot{x}_1(t) = x_2(t),$$

$$\dot{x}_2(t) = u(t) - \alpha x_2(t) - \beta x_1(t) - \gamma x_1^3(t).$$

If we choose $\alpha = 0.4, \beta = -1.1, \gamma = 1, \omega = 1.8$, it is seen that the solution trajectories of Duffing equation display complex phenomena (see Fig. 2 and Fig. 3). The figures Fig. 2 and Fig. 3 also show different periodic and chaotic dynamics when the input parameter $k$ is varied. It is the effect of $k$ that the system dynamic will close to a periodic motion when the value of $k$ is getting large.

Fig. 1 Magneto-Elastic Mechanical System

Fig. 2 (a) The phase plane of system (2) with $k=2.1$.

Fig. 2 (b) The time response of system (2) with $k=2.1$.

Fig. 3 (a) The phase plane of system (2) with $k=7$.

Fig. 3 (b) The time response of system (2) with $k=7$.

### III. The genetic algorithms

GA is a parallel and global stochastic search technique, starting with an random initial set of solutions, which is called population. Each individual in the population is called a chromosome. In this paper, a chromosome is represented by a binary bit string. The chromosomes evolve through successive iterations which are called generations in GA. During each generation, the chromosomes are evaluated by using some measures of fitness functions [11]. The new chromosomes in next generation, called offspring, are generated by crossover rule. Fitter chromosomes have higher probabilities are selected to execute crossover operation. After several generations, the
algorithms converge to the best chromosome which will reaches the optimal or near-optimal solution to the problem. Hence, the implementation of GA consists of three main operations [12-14]:

A · Reproduction

In this paper, the roulette wheel selection is used to reproduce the better chromosome. The one, which get the largest area (highest probability) in the wheel, is the best chromosome. It will be used as the parent generation for the genetic reproduction. The probability of each chromosome can be calculated by the formula:

\[
\text{prob}(P_i) = \frac{\text{fitness}(P_i)}{\sum_{i=1}^{n} \text{fitness}(P_i)}
\]

where \( n \) is population size, \( P_i \) represents the \( i \)th chromosome, and \( \text{fitness}(\cdot) \) is a fitness function of its argument.

B · Crossover

The crossover operation used in this paper is a randomly selected single point crossover. For example, for the given two chromosomes \( P_1 \) and \( P_2 \) in the following, the mating points after 5th gene are (randomly) selected for the crossover:

\[
P_1: 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \\
P_2: 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1
\]

After the crossover operation, we have the following result:

\[
P_1': 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \\
P_2': 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0
\]

C · Mutation

In order to obtain a better solution, the mutation will be performed in the last step of creating a new generation. The probability of mutation is usually in the range from 1% to 8%. The probability 8% of mutation means the average number of chromosomes used to mutation is 8 out of 100 chromosomes. The process of mutation is to choose randomly one bit from the binary string of a chromosome and then take 1's complement.

Before mutation:

\[
1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1
\]

After mutation:

\[
1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1
\]

IV. The sandwich-like algorithm

In this section, combining GA and SDM, we propose a new sandwich-like algorithm to identify the chaotic systems by BPN. Moreover, the simulation results, comparing to various algorithms, are also shown in next section to illustrate the advantage of the proposed algorithm.

The SDM is used to train the weight and bias of a neural network for a long time. Although this traditional method could obtain a fast speed of convergence in learning, but it appears the shortage of the fact that a local optimum is always found. Recently, the GA are used to replace the SDM on the training of neural network. This is due to the GA always reaches the global optimum instead of local optimum of the cost function. However it take much more time on the process of generation reproduction and then the convergent rate becomes slower than SDM. Consequently, it is obvious that the learning performance of SDM and GA are complement for each other. This causes the motivation of integrating SDM and GA as a sandwich-like algorithm (Fig. 4) for the training of a neural network.

There are three stages in sandwich-like algorithm. In the following, the function of each stage is described and a diagram for the illustration of the three stages are depicted in Fig. 5.

![Fig. 4 The three stages of sandwich-like algorithm](image)

(1) Stage 1

The first stage is to search, by steepest descent method with a set of random initial values, a set of more “nice” initial values of the weights in neural network. This set of more “nice” initial values are then the initial values of the GA in the next stage.

(2) Stage 2

Based on the initial values obtained from the first stage, the genetic algorithm is then used to make a global search of the weights which optimize the cost function of the output of neural network.

(3) Stage 3

In the final stage, for speeding up the convergent rate of the learning algorithm, the steepest descent method in used again to search the final optimal solution of weights.

![Fig. 5 The illustration of sandwich-like algorithm](image)
It can be seen that the SDM is used to speed up the learning performance of GA at the head (a set of more “nice” initial values) and the tail (a quickly converge from near-optimal to optimal) of sandwich-like algorithm. In the next section, we will show that our algorithm can obtain a better learning performance than those of other algorithm.

V. Simulation results

In this section, the results of the proposed algorithm are compared to those of existing algorithm in other researches. The chaotic systems, duffing equation and logistic map, are considered for the simulation. In the neural network, a hyperbolic tangent function (Fig. 6) is chosen as the activation function of hidden layer and a linear function (Fig. 7) is chose as the activation function of output layer.

![Hyperbolic tangent function](image)

**Fig. 6** Hyperbolic tangent function

![Linear activation function](image)

**Fig. 7** Linear activation function

Simulation results for the identification of duffing equation

The duffing equation (2) is considered for the simulation in this section. The series-parallel identification model (SPM) is used here for the prediction of duffing equation. In the SPM, the output of the plant is fed back into the identification model as shown in Fig. 8. The back-propagation neural network consists of four-layers structure is chosen for this simulation. The architecture of the BPN is of the form 3-4-2-1 (Fig. 9).

In the following, in order to compare the performance of various learning algorithm, the simulation result of SDM will first be presented. Then, different type of combination from GA and SDM will be used to train the neural network for the identification of duffing equation. The simulation results will also be shown later. Table 1 shows the necessary parameters for the simulation of duffing equation by GA. In Table 2, the comparison of training times of the neural network trained by SDM and the sandwich-like algorithm are listed corresponding to various expected MSE. It can be seen that when the expected MSE is larger than 0.0052, the SDM can train the neural network faster than sandwich-like algorithm. However, if the expected MSE is smaller than 0.0045, the training of neural network by SDM is fail (in this case the training time is 1048.32 minutes and the MSE is 0.00491) while the sandwich-like algorithm still work. This is due to the fact that the SDM always get a local optimal solution. Consequently, the sandwich-like algorithm proposed in this paper can give a more precise solution than SDM.

![Series-Parallel Identification Model](image)

**Fig. 8** Series-Parallel Identification Model

![BPN for the identification of duffing equation.](image)

**Fig. 9** BPN for the identification of duffing equation.

**Table 1** The parameters of GA and duffing equation

<table>
<thead>
<tr>
<th>Duffing Equation’s parameter</th>
<th>α</th>
<th>β</th>
<th>γ</th>
<th>ω</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>β</td>
<td></td>
<td>-1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>γ</td>
<td>1</td>
<td></td>
<td></td>
<td>1.8</td>
</tr>
<tr>
<td>ω</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
training $u(t) = 1.8 \cos(1.8 t)$

<table>
<thead>
<tr>
<th>GA’s parameter</th>
<th>Population size</th>
<th>Bit length</th>
<th>Crossover rate</th>
<th>Mutation rate</th>
<th>Code and encode region</th>
<th>Training pattern</th>
<th>Training times (duffing equation)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50</td>
<td>35</td>
<td>0.9</td>
<td>0.1</td>
<td>-1 ~ +1</td>
<td>1000</td>
<td>SDM (Minutes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sandwich-like algorithm (Minutes)</td>
</tr>
<tr>
<td>Expected MSE</td>
<td>0.0243</td>
<td>74.22</td>
<td>1.34</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0173</td>
<td>193.92</td>
<td>7.88</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0052</td>
<td>393.42</td>
<td>14.98</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.0045</td>
<td>792.42</td>
<td>Fail (1048.32 minutes with MSE: 0.00491)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2 Comparison of training times (duffing equation)

### VI. Conclusion

This paper presents a sandwich-like algorithm for the training of back-propagation neural network. The chaotic systems, the duffing equation, are identified by the neural network. The sandwich-like algorithm integrates GA and SDM to get a faster learning speed than GA and a more precise result than SDM. According to this algorithm, we trained the initial values by using SDM. Then the GA is used for global search of optimization. Finally, for speeding up the convergent rate of learning, the approximate optimum was found out by SDM again. The simulation results show that sandwich-like algorithm is much more effective then those of other algorithms.

### Acknowledgement

This research work was supported by the National Science Council of the Republic of China under contract NSC 91-2213-E-366-005-. Moreover, the author would like to thank Shih-Hung Chiu for his helps in the implementation of the proposed algorithm.

### References


