On Interpretation of Graffiti Commands for eBooks using a Neural Network and an Improved Genetic Algorithm[1]

H.K. Lam, S.H. Ling, K.F. Leung, and F.H.F. Leung
Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

Abstract

This paper presents the interpretation of graffiti commands for Electronic Books (eBooks). The interpretation process is achieved by training a proposed neural network (NN) with link switches using an improved genetic algorithm (GA). By introducing the switches to the links, the proposed NN can learn the optimal network structure automatically. The structure and the parameters of the NN are tuned by the improved GA, which is implemented by floating point numbers. The processing time of the improved GA is shorter as reflected by some benchmark test functions. Simulation results on interpreting graffiti commands for eBooks using the proposed NN with link switches and the improved GA will be shown.

I. Introduction

Notebook Computers and Personal Digital Assistants (PDAs) are modifying our life. One of the changes is our reading habit. Electronic Books (eBooks) are winning their popularity as a kind of media that can offer rich contents and features such as multimedia effects, instant dictionaries and bookmark functions etc. within a small handheld device. An eBook Reader should have no keyboard or mouse. The main input device is a touch screen. As many functions are implemented in a single eBook Reader, it is not convenient to access these functions through menus without hot keys. However, even when hot keys (as icons on the screen) are employed, time is needed to access them, especially when the number of hot keys is large. Thanks to the touch screen, a one-step commanding process using graffiti is proposed for eBooks. For instance, when a user draws a straight line on the screen, the eBook Reader is able to respond to the command represented by the straight line. However, computers are only good at numerical, but not symbolic, manipulation, while the interpretation of graffiti commands is symbolic manipulation process. Thus, a way to convert a symbolic manipulation process to a numerical manipulation process should be found. In this paper, a neural network (NN) with link switches is proposed to perform the interpretation of graffiti commands. An improved GA will be developed to train the proposed NN.

GA is a powerful directed random search technique invented by Holland [1] in 1975 to handle optimization problems [1-2, 5]. This is especially useful for complex optimization problems with a large number of parameters that make global analytical solutions difficult to obtain. It has been widely applied in different areas such as fuzzy control [9-11, 15], path planning [12], greenhouse climate control [13], modeling and classification [7-8, 14] etc..

[1] The work described in this paper was substantially supported by a Research Grant of The Hong Kong Polytechnic University (project number A432).

NNs have been proved to be a universal approximator [16]. A 3-layer feed-forward NN can approximate any nonlinear function to an arbitrary accuracy. NNs were successfully applied in areas such as prediction [7], system modeling and control [16]. In view of its specific structure, a NN can be used to realize a learning process [2]. In general, the learning steps are as follows. First, a network structure is defined. Second, an algorithm is chosen for realizing the learning process. Usually, the structure of the NN is fixed for a learning process. However, this fixed structure may not provide the best performance within a given training period. If the NN structure is too complicated, the training period will be long and the implementation cost will be high.

The contributions of this paper are six-fold. First, new genetic operators are introduced to the improved GA [3-4, 6, 17]. Second, the improved GA is implemented in floating-point number; hence, the processing time is shorter [1-2, 5]. Third, one parameter (the population size), instead of three, is needed for tuning. This makes the improved GA simple and easy to use. Fourth, a three-layer NN with a switch introduced in each link is proposed to facilitate the tuning of the optimal network structure in terms of cost, testing time and processing time. Fifth, the improved GA can help tuning the structure and the parameters of the proposed NN. Sixth, the proposed NN is able to interpret graffiti commands for eBooks after it has been trained by the improved GA. It will be shown that the proposed NN trained by the improved GA performs better than a traditional feed-forward NN [2] trained by the traditional GA [1-2, 5].

II. Improved Genetic Algorithm

Genetic algorithms (GAs) [1] are powerful searching algorithms to handle optimization problems. In this paper, the traditional GA is modified and new genetic operators are introduced to improve its performance. The probabilities of crossover and mutation in the traditional GA are no longer needed. Only the population size has to be defined. The improved GA process is shown in Fig. 1.

A. *Initial Population*

The initial population is a potential solution set P. The first set of population is usually generated randomly.

$$P = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_{pop\_size}\}, \tag{1}$$

$$\mathbf{p}_i = \begin{bmatrix} p_{i_1} & p_{i_2} & \cdots & p_{i_j} & \cdots & p_{i_{no\_vars}} \end{bmatrix}, \tag{2}$$

$$para^j_{\min} \le p_{i_j} \le para^j_{\max},\ i = 1, \ldots, pop\_size;\ j = 1, \ldots, no\_vars \tag{3}$$

where pop_size denotes the population size; no_vars denotes the number of variables to be tuned; p_{i_j} are the parameters to be tuned; $para^j_{\min}$ and $para^j_{\max}$ are the minimum and maximum values of the parameter p_{i_j}. It can be seen from (1) to (3) that the potential solution set P contains some candidate solutions $\mathbf{p}_i$ (chromosomes). The chromosome $\mathbf{p}_i$ contains some variables p_{i_j} (genes).

B. Evaluation

Each chromosome in the population will be evaluated by a defined fitness function. The better chromosomes will return higher values in this process. The fitness function to evaluate a chromosome in the population can be written as,

$$fitness = f(\mathbf{p}_i) \qquad (4)$$

The form of the fitness function depends on the application.

C. Selection

Two chromosomes in the population will be selected to undergo genetic operations for reproduction. The chromosome having a higher fitness value should have a higher chance to be selected. The selection can be done by assigning a probability q_i to the chromosome $\mathbf{p}_i$:

$$q_i = f(\mathbf{p}_i) \Big/ \sum_{j=1}^{pop\_size} f(\mathbf{p}_j),\ i = 1, 2, \ldots, pop\_size \qquad (5)$$

The cumulative probability $\hat{q}_i$ for $\mathbf{p}_i$ is defined as,

$$\hat{q}_i = \sum_{j=1}^{i} q_j,\ i = 1, 2, \ldots, pop\_size \qquad (6)$$

The selection process randomly generates a nonzero floating-point number, $d \in [0 \ \ 1]$, for each chromosome. Then, the chromosome $\mathbf{p}_i$ is chosen if $\hat{q}_{i-1} < d \le \hat{q}_i$, i = 1,2, …, *pop_size*, and $\hat{q}_0 = 0$. It can be observed from this selection process that a chromosome having a larger $f(\mathbf{p}_i)$ will have a higher chance to be selected.

D. Genetic Operations

The genetic operations are the averaging and the mutation operations. If two selected chromosomes are $\mathbf{p}_1$ and $\mathbf{p}_2$, the offspring generated by the averaging process is given by,

$$\mathbf{os} = \begin{bmatrix} os_1 & os_2 & \cdots & os_{no\_vars} \end{bmatrix} = (\mathbf{p}_1 + \mathbf{p}_2)/2 \qquad (7)$$

This offspring (7) will then undergo the mutation operation. Three new offspring will be generated:

$$\mathbf{nos}_j = \begin{bmatrix} os_1^j & os_2^j & \cdots & os_{no\_vars}^j \end{bmatrix} + \begin{bmatrix} b_1 \Delta nos_1 & b_2 \Delta nos_2 & \cdots & b_{no\_vars} \Delta nos_{no\_vars} \end{bmatrix},\ j = 1, 2, 3 \qquad (8)$$

where b_i, $i = 1, 2, \ldots, no\_vars$, can only take the value of 0 or 1, Δnos_i, i = 1, 2, …, *no_vars*, are randomly generated floating-point numbers with $para_{\min}^i \le os_i^j + \Delta nos_i \le para_{\max}^i$. The first new offspring (j = 1) has only one b_i = 1 (i being randomly generated within the range) and all the others are zeros. The second new offspring has some b_i randomly set to 1 and others are zeros. The third new offspring has all b_i = 1. These three new offspring will then be evaluated using the fitness function of (4). The one with the largest fitness value f_l will replace the chromosome with the smallest fitness value f_s in the population if $f_l > f_s$. After selection, averaging, and mutation, a new population is generated. This new population will repeat the same process. Such an iterative process can be terminated when the result reaches a defined condition, e.g., the change of the fitness values between the current and the previous iteration is less than 0.001.

III. Neural Network with Link Switches and Tuning

By introducing a switch to each link, not only the parameters but also the structure of the NN can be tuned using the improved GA.

A. Neural Network with Link Switches

NNs [5] for tuning usually have a fixed structure large enough to fit a given application. This often increases the implementation cost. In this section, a three-layer multiple-input-multiple-output NN is proposed which facilitates not only the tuning of the network parameters but also the network structure. The proposed 3-layer network is shown in Fig. 2. The main different point is that a unit step function is introduced to each link:

$$\delta(\alpha) = \begin{cases} 0 \text{ if } \alpha < 0 \\ 1 \text{ if } \alpha \ge 0 \end{cases},\ \alpha \in \Re \qquad (9)$$

This is equivalent adding a switch to each link of the NN. From Fig. 2, the input-output relationship is given by

$$y_k(t) = \sum_{j=1}^{n_h} \delta(s_{jk}^2) w_{jk}\, logsig\left[\sum_{i=1}^{n_{in}} \left(\delta(s_{ij}^1) v_{ij} z_i(t) - \delta(s_j^1) b_j^1 \right) \right] - \delta(s_k^2) b_k^2,$$
$$k = 1, 2, \ldots, n_{out} \qquad (10)$$

$z_i(t)$, $i = 1, 2, \ldots, n_{in}$, are inputs which are functions of a variable t; n_{in} denotes the number of input; v_{ij}, $i = 1, 2, \ldots, n_{in}$; $j = 1, 2, \ldots, n_h$, denote the weight of the link between the i-th input and the j-th hidden node. n_h denotes the number of hidden nodes; s_{ij}^1, $i = 1, 2, \ldots, n_{in}$; $j = 1, 2, \ldots, n_h$, denotes the parameter between the link from the i-th input to the j-th hidden node. s_{jk}^2, $j = 1, 2, \ldots, n_h$; $k = 1, 2, \ldots, n_{out}$, denotes the switch between the link from the j-th hidden node to the k-th output; n_{out} denotes the number of outputs of the NN; b_j^1 and b_k^2 denote the biases for the hidden nodes and output nodes respectively; s_j^1 and s_k^2 denote the switches of the biases of the hidden and output layers respectively; $logsig(\cdot)$ denotes the logarithmic sigmoid function:

$$logsig(\alpha) = \frac{1}{1 + e^{-\alpha}},\ \alpha \in \Re \qquad (11)$$

$y_k(t)$, $k = 1, 2, \ldots, n_{out}$, denotes the k-th output of the proposed NN. By introducing the switches, the weights v_{ij} and the switch states can be tuned. In can be seen that the weights of the links govern the input-output relation of the NN while the switches of the links govern the structure of the NN.

B. Tuning of the Parameters and Structure

In this section, the proposed NN is employed to learn the input-output relationship of an application using the improved GA. The input-output relationship is described by,

$$\mathbf{y}^d(t) = \mathbf{g}\left(\mathbf{z}^d(t)\right),\ t = 1, 2, \ldots, n_d \qquad (12)$$

where $\mathbf{y}^d(t) = \begin{bmatrix} y_1^d(t) & y_2^d(t) & \cdots & y_{n_{out}}^d(t) \end{bmatrix}$ and $\mathbf{z}^d(t) = \begin{bmatrix} z_1^d(t) & z_2^d(t) & \cdots & z_{n_{in}}^d(t) \end{bmatrix}$ are known inputs and outputs of an unknown function $\mathbf{g}(\cdot)$ respectively. n_d is the number of data pairs. The fitness function is defined as,

$$fitness = \frac{1}{1+err} \tag{13}$$

$$err = \sum_{k=1}^{n_{out}} \frac{n_k \sum_{t=1}^{n_d} \left| y_k^d(t) - y_k(t) \right|}{n_d} \tag{14}$$

The objective is to maximize the fitness value of (13) using the improved GA by setting the chromosome to be $\left[s_{jk}^2 \quad w_{jk} \quad s_{ij}^1 \quad v_{ij} \quad s_j^1 \quad b_j^1 \quad s_k^2 \quad b_k^2 \right]$ for all i, j, k. Refer to (14), the maximum fitness value may be dominated by the error value of certain outputs, which may not be of main interest. To avoid this, n_k, k = 1, 2, ..., n_d, are chosen to reduce the effects of the error from these dominated outputs.

IV. Application Results

The interpretation of graffiti commands for eBooks will be presented. A point on the eBook screen is characterized by a number. The interpretation process is achieved by a 3-layer NN (8-input-single-output) with link switches. The 8 inputs nodes, z_i, i = 1, 2, ... ,8, represent 8 sample points of the graffiti. These 8 sample points are taken uniformly from the input graffiti. The output node $y(t)$ represents the graffiti value. We define 3 graffiti commands: rectangle for opening the bookmark application, triangle for playing movie and straight line for going to the next page. The training input data, $\mathbf{z}^d(t)$, are 8 sets of 8 sample points for each graffiti. Hence, we have 8 × 3 = 24 sets of 8 sample points for the 3 graffiti commands. $y(t) \in [0.1 \quad 0.3)$ represents a straight line, $y(t) \in [0.3 \quad 0.5)$ represents a triangle, $y(t) \in [0.5 \quad 0.7)$ represents a rectangle. The training output data, $\mathbf{y}^d(t)$, takes the middle value. Thus, $\mathbf{y}^d(t)$ = 0.2 for straight line, 0.4 for triangle and 0.6 for rectangle. The number of hidden nodes is 11. Referring to (10), the proposed NN used for the interpretation process is governed by,

$$y_1(t) = \sum_{j=1}^{11} \delta(s_{j1}^2) w_{j1} logsig\left[\sum_{i=1}^{24} \left(\delta(s_{ij}^1) v_{ij} z_i(t) - \delta(s_j^1) b_j^1 \right) \right] - \delta(s_1^2) b_1^2 \tag{15}$$

$$fitness = \frac{1}{1+err} \tag{16}$$

$$err = \sum_{t=1}^{24} \frac{\left| y_1^d(t) - y_1(t) \right|}{24} \tag{17}$$

The improved GA is used to tune the parameters and structure of the NN of (15) by maximizing the fitness function of (16). The best fitness value is 1 and the worst one is 0. The population size is 10. The lower and the upper bounds of the link weights are defined as $\frac{-3}{\sqrt{n_h}} \ge v_{ij}, w_{jk}, b_j^1, b_1^2 \ge \frac{3}{\sqrt{n_h}}$ and, $-1 \ge s_{j1}^2, s_{ij}^1, s_j^1, s_1^2 \ge 1, i = 1, 2, \ldots, 3; j = 1, 2, \ldots, n_h, k = 1$ [18]. The chromosomes used for the improved GA are $\left[s_{j1}^2 \quad w_{jk} \quad s_{ij}^1 \quad v_{ij} \quad s_j^1 \quad b_j^1 \quad s_k^2 \quad b_1^2 \right]$. The initial values of the link weights are randomly generated. For comparison, a fully connected 3-layer (8-input-single-output) feed-forward NN [2] trained by traditional GA is used for interpreting graffiti commands. The working conditions are kept the same. The parameter coding has a bit length of 10. The probabilities of the crossover and mutation are 0.65 and 0.05 respectively.

The number of iterations to train the two NNs is 1000. After training, 4 graffiti samples of each kind of graffiti (4 × 3 = 12 testing graffiti commands) are used to test the performance of trained NNs. The simulation results are tabulated in Table I. It can be observed that the proposed NN trained with the improved GA provides better results in terms of the accuracy (fitness values), the number of links, training and testing mean absolute errors (MAE) and the training time. The number of connected link is 60 after learning instead of 111, which includes the bias links. Fig. 3 and Fig. 4 show the output values of the two NNs for the training and testing graffiti respectively. In Fig. 3, training patterns 1 to 8 are graffiti in rectangle, patterns 9-16 are graffiti in triangle and patterns 17-24 are graffiti in straight line. From this figure, we can see that all output values are within the defined region and the interpretation is successful. In Fig. 4, testing patterns 1 to 4 are graffiti in rectangle, patterns 5-8 are graffiti in triangle and patterns 9-12 are graffiti in straight line. Again, all output values are within the defined regions when the proposed NN is used. However, if the traditional NN is used, pattern 2 is out of the defined region.

V. Conclusion

An improved GA has been proposed in this paper. It is implemented by floating-point numbers. As no coding and encoding of the chromosomes are necessary, the time for learning is shorter. New operators have been introduced to the improved GA. By introducing a switch to each link, a NN that facilitates the tuning of its structure has also been proposed. Using the improved GA, the proposed NN will not only learn the input-output relationship of an application but also the optimal network structure. A fully connected NN will become a partially connected NN after learning. This implies a lower cost of implementation. The proposed NN with link switches trained by the improved GA has been applied to interpret graffiti commands for eBooks. It has been shown that our proposed network can offer a better result.

References

[1] J.H. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, MI, 1975.

[2] D.T. Pham and D. Karaboga, *Intelligent Optimization Techniques, Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*, Springer, 2000.

[3] Y. Hanaki, T. Hashiyama, and S. Okuma, "Accelerated evolutionary computation using fitness estimation," in *Proc. IEEE SMC '99 Conference*, vol. 1, 1999, pp. 643-648.

[4] K.A. de Jong, "An analysis of the behavior of a class of genetic adaptive systems," *Ph.D. Thesis*, University of Michigan, Ann Arbor, MI., 1975.

[5] Z. Michalewicz, *Genetic Algorithm + Data Structures = Evolution Programs*, second, extended edition, Springer-Verlag, 1994.

[6] G.X. Yao and Y. Liu "Evolutionary programming made

faster," *IEEE Trans., on Evolutionary Computation*, vol. 3, no. 2, pp.82-102, July 1999

[7] M. Li, K. Mechrotra, C. Mohan, and S. Ranka, "Sunspot numbers forecasting using neural network," in *Proc. 5th IEEE International Symposium on Intelligent Control, 1990*, pp.524-528.

[8] T.J. Cholewo, J.M. Zurada, "Sequential network construction for time series prediction," in *Proc. Int. Conf. on Neural Networks*, 1997, vol.4, pp.2034-2038.

[9] B.D. Liu, C.Y. Chen and J.Y. Tsao, "Design of adaptive fuzzy logic controller based on linguistic-hedge concepts and genetic algorithms," *IEEE Trans. Systems, Man and Cybernetics, Part B*, vol. 31 no. 1, pp. 32-53, Feb. 2001.

[10] Y.S Zhou and L.Y, Lai "Optimal design for fuzzy controllers by genetic algorithms," *IEEE Trans., Industry Applications*, vol. 36, no. 1, pp. 93-97, Jan.-Feb. 2000.

[11] C.F. Juang, J.Y. Lin and C.T. Lin, "Genetic reinforcement learning through symbiotic evolution for fuzzy controller design," *IEEE Trans., Systems, Man and Cybernetics, Part B*, vol. 30, no. 2, pp. 290–302, April, 2000.

[12] H. Juidette and H. Youlal, "Fuzzy dynamic path planning using genetic algorithms," *Electronics Letters*, vol. 36, no. 4, pp. 374-376, Feb. 2000

[13] R. Caponetto, L. Fortuna, G Nunnari, L. Occhipinti and M. G Xibilia, "Soft computing for greenhouse climate control," *IEEE Trans., Fuzzy Systems*, vol. 8, no. 6, pp. 753-760, Dec. 2000.

[14] M. Setnes and H. Roubos, "GA-fuzzy modeling and classification: complexity and performance," *IEEE. Trans, Fuzzy Systems*, vol. 8, no. 5, pp. 509–522, Oct. 2000.

[15] K. Belarbi, F. Titel, "Genetic algorithm for the design of a class of fuzzy controllers: an alternative approach," *IEEE Trans Fuzzy Systems*, vol. 8, no. 4, pp. 398-405, Aug. 2000.

[16] M. Brown and C. Harris, *Neuralfuzzy adaptive modeling and control.* Prentice Hall, 1994.

[17] S. Amin and J.L. Fernandez-Villacanas, "Dynamic local search," in *Proc. 2nd Int. Conf. Genetic Algorithms in Engineering Systems: Innovations and Applications*, 1997, pp129-132.

[18] L.F.F. Wessels and E. Barnard, "Avoiding false local minima by proper initialization of connections," *IEEE Trans. Neural Network*, vol. 3, no.6, pp. 899-905, 1992.

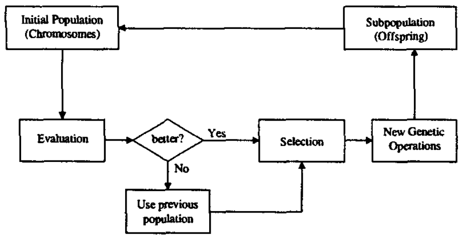

Fig. 1. Improved GA.

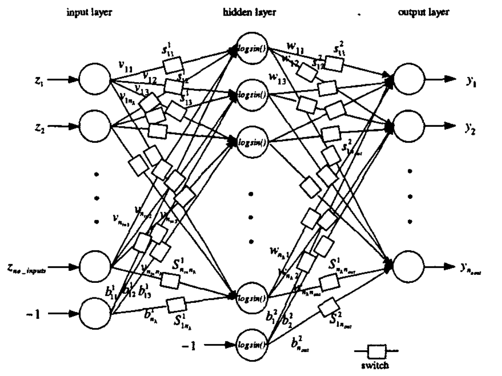

Fig. 2. The proposed 3-layer NN with line switches.

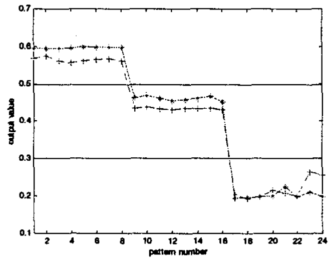

Fig. 3. The output values of the two NNs for the 24 training graffiti; Solid line with (*): the proposed NN; Dash line with (+): the traditional NN.

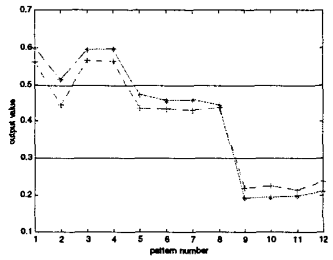

Fig. 4. The output values of the two NNs for the 12 testing graffiti; Solid line with (*): the proposed NN; Dash line with (+): the traditional NN.

TABLE I
RESULTS OF THE PROPOSED AND THE TRADITIONAL NN FOR INTERPRETING GRAFFITI COMMANDS AFTER TRAINING FOR 1000 ITERATIONS.

	Fitness value	No. of links	Training error	Testing error	Searching Time (s)
Proposed	0.977343	60	0.0232	0.0306	494.44
Traditional	0.971251	111	0.0296	0.0408	1857.5