JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 21, 721-732 (2005)

Short Paper

Geometrical Perspective on Learning Behavior*

CHENG-YUAN LIOU, JAU-CHI HUANG AND YEN-TING KUO
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, 106 Taiwan
E-mail: cyliou@csie.ntu.edu.tw

This paper constructs a geometrical perspective to justify the slow learning period and fast learning period during training. It plots the error surfaces and the solution spaces in the input space for a single neuron with two inputs. It records various training paths in this space using the back-propagation (BP) training algorithm [6]. It finds the relations between the learning curve and training path.

Keywords: multilayer network, back-propagation learning, neural network, momentum method, premature saturation

1. INTRODUCTION

Since strong assumptions and approximations are used to achieve premature saturation (see [3]), the results obtained are not precise or clear. This work provides a new way to view exquisite learning behavior. The error surface, or energy surface, and the solution space for descending learning rules (BP) [6, 9] have commonly been plotted in the weight space or in hypercube space [1, 4]. Instead of this common approach, we draw the error surface and the solution space in the input space [5]. This provides a very different viewpoint on the training paths. Consider a neuron with weights [w_1, w_2, w_3] and two inputs {x_1, x_2}, plus a fixed input of 1 for the threshold w_3 (see Fig. 1 (a)). In Fig. 1 (b), L designates a decision line given by weights [w_1, w_2, w_3].

By solving the joint point X, (a_1, a_2), between the decision line L: $w_1x_1 + w_2x_2 + w_3 = 0$ and line L': $w_2x_1 - w_1x_2 = 0$, which is perpendicular to line L and passes through the origin, we obtain the following equations:

$$\frac{w_1}{w_3} = \frac{a_1}{a_1^2 + a_2^2}, \quad \frac{w_2}{w_3} = \frac{a_2}{a_1^2 + a_2^2}. \tag{1}$$

Received August 28, 2003; revised May 14, 2004; accepted January 6, 2005.
Communicated by Chin-Teng Lin.
* This paper was supported in part by the National Science Council of Taiwan, R.O.C., under project NSC 93-2213-E-002-081.

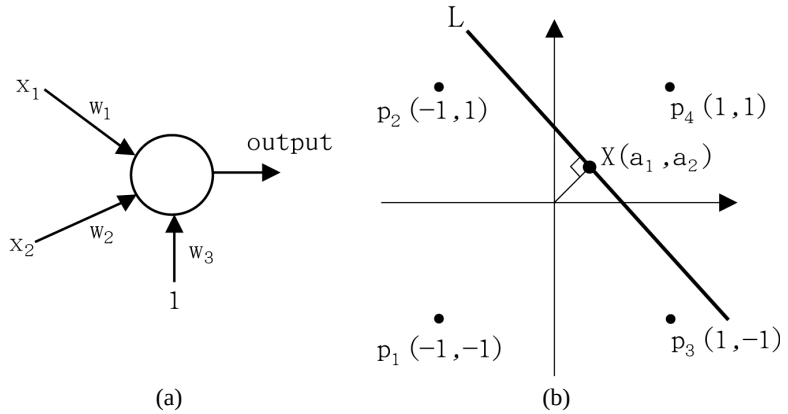


Fig. 1. (a) A single neuron diagram; (b) The decision line L.

Line L can be represented by a single perpendicular point X at location (a_1, a_2). Reformulating the above equations, we obtain

$$a_1 = \frac{-w_1 w_3}{w_1^2 + w_2^2},\ a_2 = \frac{-w_2 w_3}{w_1^2 + w_2^2}. \tag{2}$$

We will use this decision point, (a_1, a_2), to represent the whole decision line L (or decision hyperplane). Since a decision line L: $w_1x_1 + w_2x_2 + w_3 = 0$ and its opposite decision line L_{-1}: $-w_1x_1 - w_2x_2 - w_3 = 0$ share the same line boundary. Accordingly, each point (a_1, a_2) corresponds to two decision lines with opposite directions. Therefore, we assign $C = \pm 1$ to represent these two opposite directions. We write the equations for these two opposite decision lines as follows:

$$C\left[\frac{a_1}{a_1^2 + a_2^2} x_1 + \frac{a_2}{a_1^2 + a_2^2} x_2 - 1\right] = 0,\ C = \pm 1. \tag{3}$$

In the next section, we will display the collections of such points (a_1, a_2) to represent error surfaces, the solution space, and training paths, respectively. We will also provide an example to illustrate the learning behaviors in detail.

2. PERSPECTIVE IN INPUT SPACE

We use the four binary patterns $\{(x_1^{(p)}, x_2^{(p)}),\ p = 1, \ldots, 4\}$ as inputs as shown in Fig. 1 (b) to construct error surfaces, the solution spaces, and training paths. The desired outputs d_p of these four patterns have $16 = 2^4$ combinations, $d_p \in \{F_i, i = 0, \ldots, 15\}$. Each combination is a Boolean function. We list them in Table 1.

Table 1. List of all the Boolean functions with two inputs $\{x_1, x_2\}$.

	$x_1^{(p)}$	$x_2^{(p)}$	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
p_1 (p = 1)	– 1	– 1	– 1	– 1	– 1	– 1	– 1	– 1	– 1	– 1	1	1	1	1	1	1	1	1
p_2 (p = 2)	– 1	1	– 1	– 1	– 1	– 1	1	1	1	1	– 1	– 1	– 1	– 1	1	1	1	1
p_3 (p = 3)	1	– 1	– 1	– 1	1	1	– 1	– 1	1	1	– 1	– 1	1	1	– 1	– 1	1	1
p_4 (p = 4)	1	1	– 1	1	– 1	1	– 1	1	– 1	1	– 1	1	– 1	1	– 1	1	– 1	1

2.1 Error Surfaces and the Solution Space

Since each decision point X represents two hyperlines (Eq. (3)), there are two errors, $E_X^{(1)}$ and $E_X^{(2)}$, at this point. They can be calculated as

$$E_X^{(i)} = \frac{1}{2}\sum_{p=1}^{4}(d_p - o_p^{(i)})^2, \; i = 1 \text{ or } 2, \tag{4}$$

$$o_p^{(i)} = \sigma(net_p^{(i)}) = 2\left(\frac{1}{1+\exp(-net_p^{(i)})} - \frac{1}{2}\right). \tag{5}$$

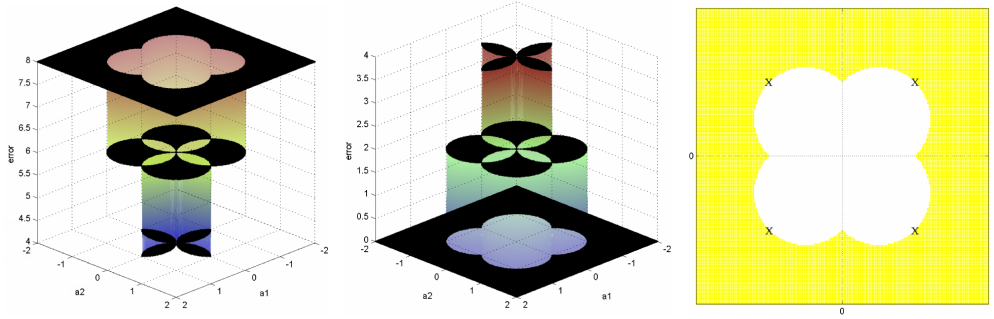
In the above equation, d_p is the desired response of the input pattern $(x_1^{(p)}, x_2^{(p)})$. The variable $net_p^{(i)}$ of the sigmoid function $\sigma(.)$ has two possible values. They are

$$net_p^{(2)} = -\,net_p^{(1)} = -\frac{a_1}{a_1^2 + a_2^2}x_1^{(p)} - \frac{a_2}{a_1^2 + a_2^2}x_2^{(p)} + 1. \tag{6}$$

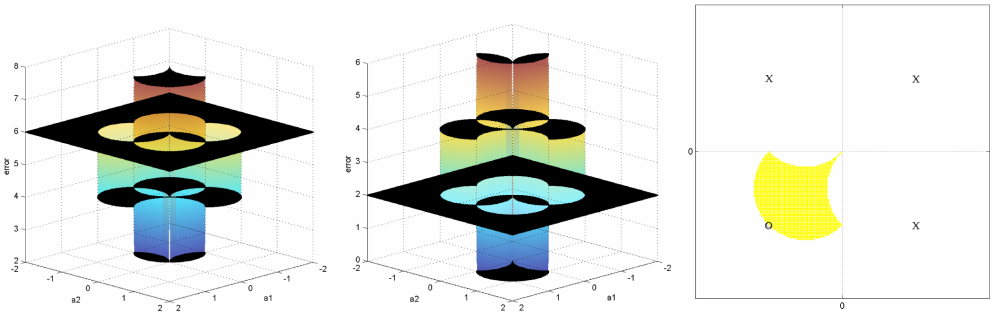
The error surfaces are continuous collections of these two kinds of errors, $E_X^{(1)}$ and $E_X^{(2)}$. The *hard-limited error surface* can be obtained by replacing the sigmoid function $\sigma(net_p^{(i)})$ with a hard-limited activation function $\text{sgn}(net_p^{(i)})$.

We plot the two hard-limited error surfaces and the solution spaces where $E_X^{(1)} = 0$ or $E_X^{(2)} = 0$ in Fig. 2. Each point in the shaded area of the solution spaces represents a correct decision line. Note that the minimal errors in Figs. 2 (1a), (2a), (3a), (4a), (5a) are not zero. There is no correct solution under the surfaces with $C = -1$, because the directions of their decision lines are wrong. Neither of the minimal errors in Figs. 2 (8a) and (8b) are zero, so the solution space does not exist (see Fig. 2 (8c)). This is called the XOR problem. Other Boolean functions that are not shown in Fig. 2 can be obtained by changing the signs of all input patterns.

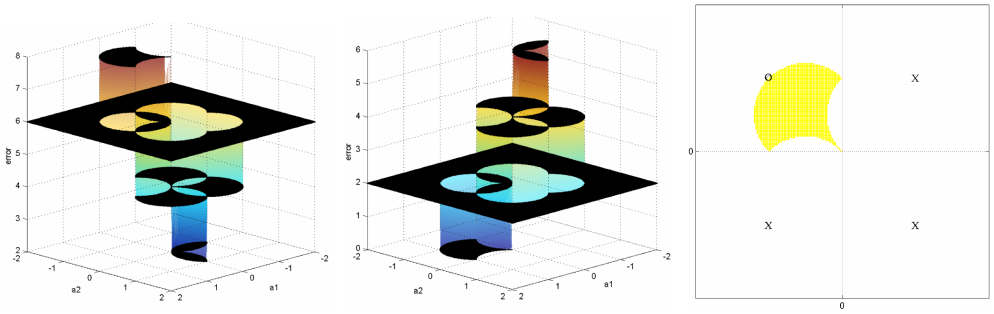
The solution space for the 14 Boolean functions (except XOR and XNOR) is shown in Fig. 3. It is useful for examining the various transition paths in this space using the BP algorithm and this will be discussed in the next section.



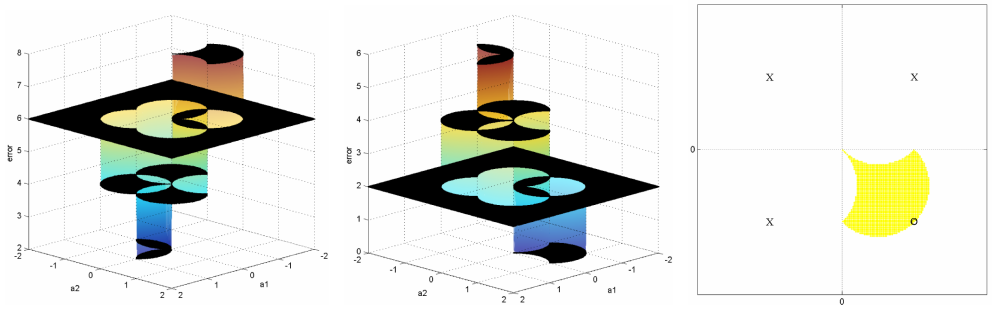
(1a) Error surface. $C = -1$. (1b) Error surface. $C = 1$. (1c) Solution space of F_0.



(2a) Error surface. $C = -1$. (2b) Error surface. $C = 1$. (2c) Solution space of F_8.

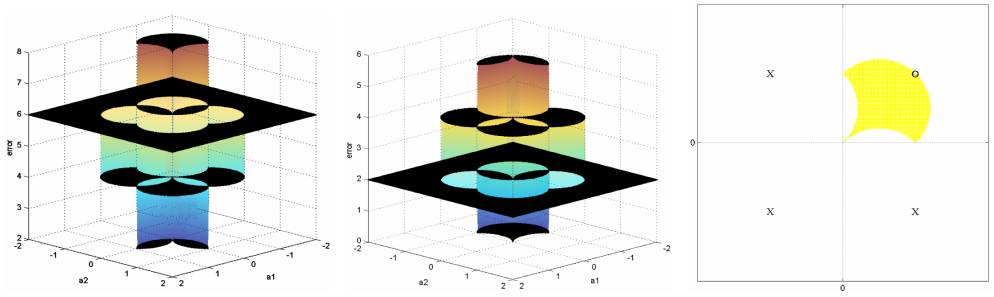


(3a) Error surface. $C = -1$. (3b) Error surface. $C = 1$. (3c) Solution space of F_4.

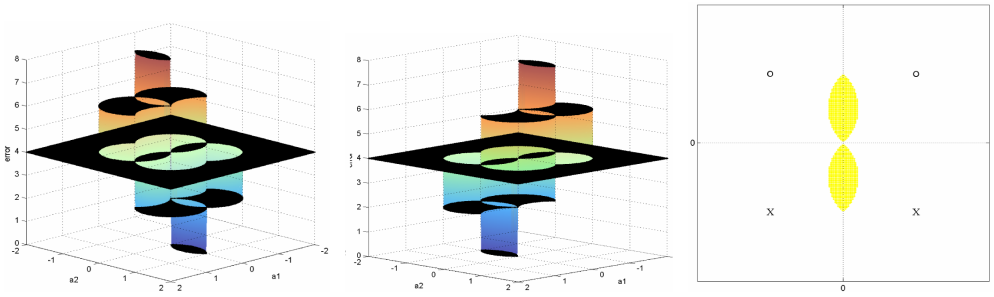


(4a) Error surface. $C = -1$. (4b) Error surface. $C = 1$. (4c) Solution space of F_2.

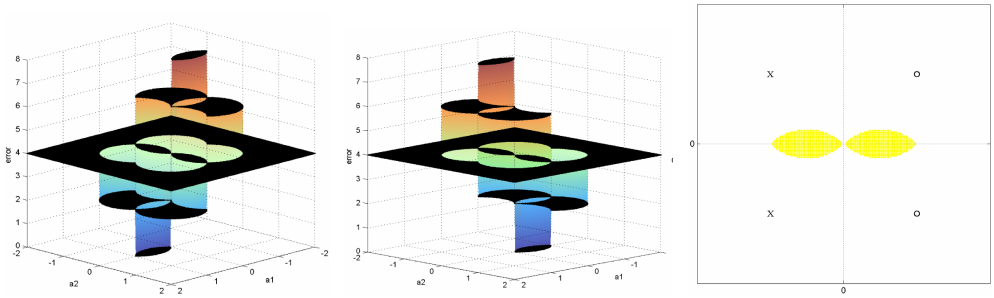
Fig. 2. The hard-limited error surfaces and the solution spaces for eight Boolean functions (F_0, F_8, F_4, F_2, F_1, F_5, F_3, F_6). Note that 'o' represents 1 and 'x' represents – 1 in the desired response.



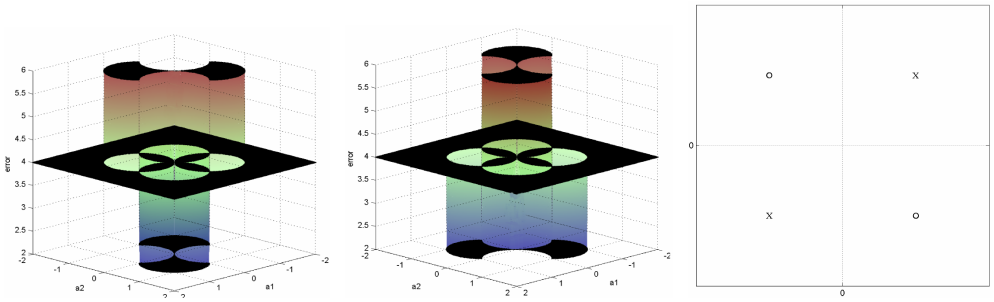
(5a) Error surface. $C = -1$. (5b) Error surface. $C = 1$. (5c) Solution space of F_1.



(6a) Error surface. $C = -1$. (6b) Error surface. $C = 1$. (6c) Solution space of F_5.



(7a) Error surface. $C = -1$. (7b) Error surface. $C = 1$. (7c) Solution space of F_3.



(8a) Error surface. $C = -1$. (8b) Error surface. $C = 1$. (8c) Solution space of F_6.

Fig. 2. (Cont'd) The hard-limited error surfaces and the solution spaces for eight Boolean functions (F_0, F_8, F_4, F_2, F_1, F_5, F_3, F_6). Note that 'o' represents 1 and 'x' represents – 1 in the desired response.

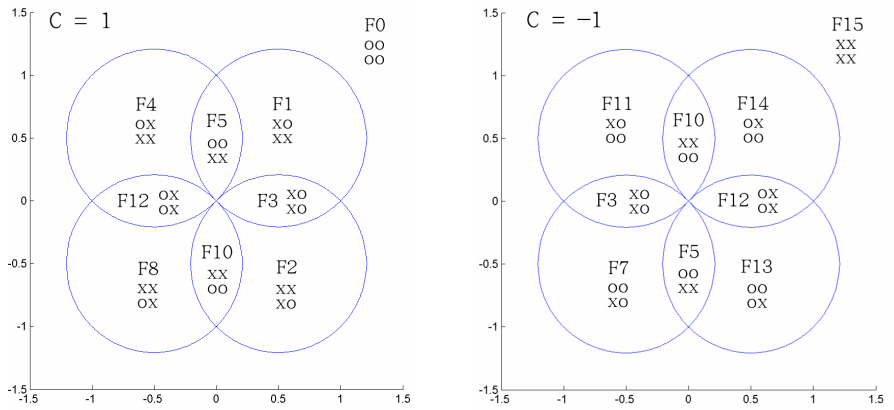
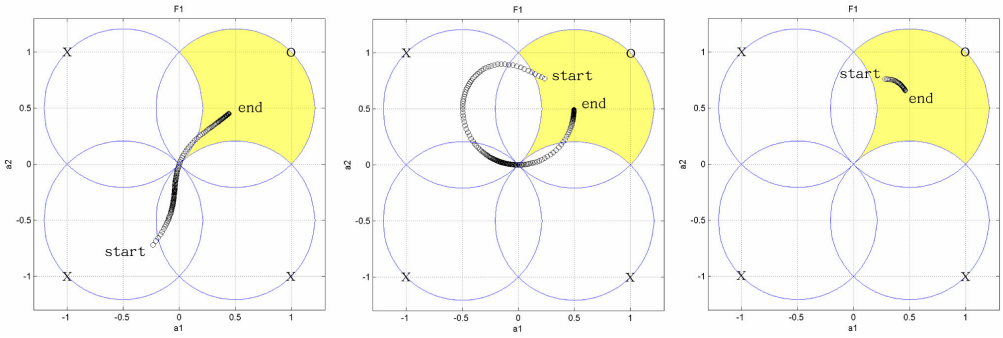


Fig. 3. The solution spaces of all 14 Boolean functions. The 'o' and 'x' marks are consistent with those in Fig. 2. Each closed area is a Boolean area.

2.2 Training Paths

We will follow a sequence of decision points, or a training path, during BP training to show important learning behavior. We use the desired output F_1 as an example. The results are shown in Fig. 4, which also shows some interesting properties.



(a) The initial weights are [0.9, 2.6, 2.2] and the converged weights are [2.0, 2.0, – 1.8].

(b) The initial weights are [– 0.9, – 2.6, 2.2] and the converged weights are [1.9, 1.9, – 1.9].

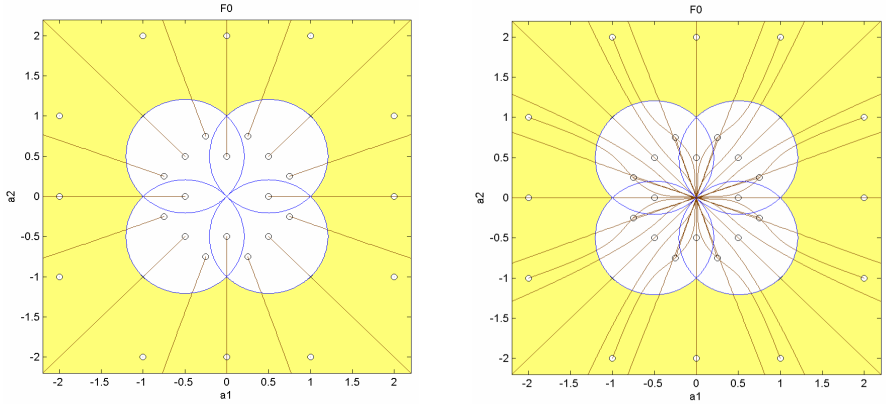
(c) The initial weights are [0.9, 2.6, – 2.2] and the converged weights are [1.9, 1.9, – 1.9].

Fig. 4. Training paths with different initial weights.

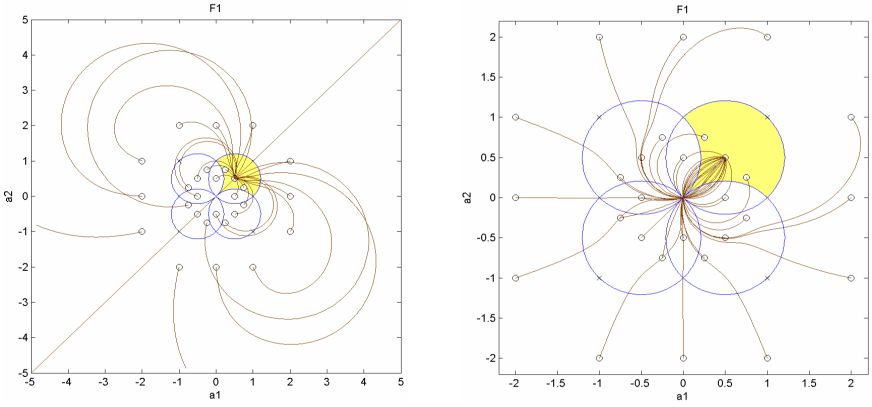
In Fig. 4 (a), the initial weights are [0.9, 2.6, 2.2], and the converged weights are [2.0, 2.0, – 1.8]. We observe that the path passes through the origin ($w_3 = 0$), where the sign of w_3 switches. In Fig. 4 (b), the initial weights are [– 0.9, – 2.6, 2.2], and the converged weights are [1.9, 1.9, – 1.9], where the signs of all weights must be changed to reach convergence. We observe that the training path first passes through the x_2 axis, where the sign of w_1 changes. Then, it changes direction and passes through the origin ($w_3 = 0$ and $w_2 = 0$) in order to switch the other two signs of w_2 and w_3 at the same time.

If the signs of the initial weights are equal to those of the converged weights, then the path will not have to pass through the origin or any axis in order to switch their signs. It will move directly to the solution (see Fig. 4 (c)).

In many cases, as shown in Figs. 4 (a), (b) and Figs. 5 (a), (b) ($C = -1$), the path will pass either through an axis or the origin, and the signs of the corresponding weights will change. The path will jump to the other surface at the origin to reach the minimum. With this property, we can initially assign very small magnitudes to the weights. This will make it easier to change the weights' signs during training. In other cases, as shown in Figs. 5 (b), (c) ($C = 1$), the path will detour and surround the origin. In Fig. 5, we plot all the different kinds of paths, where we omit some symmetric plots.

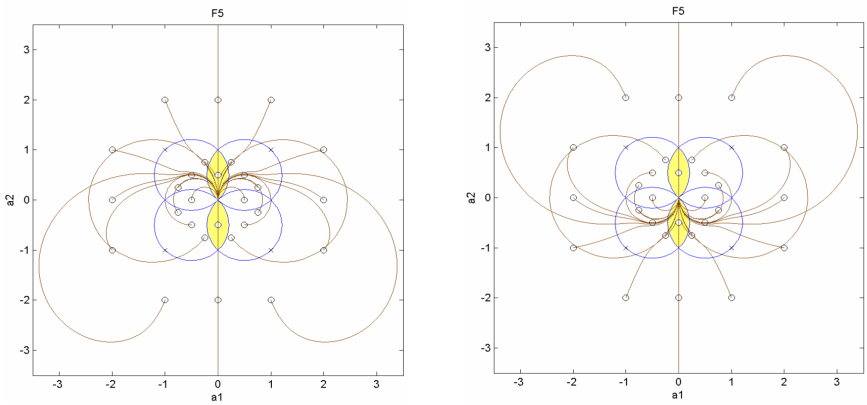


(a) The training paths for F_0. In $C = 1$, all the paths proceed outwards to the correct solution space directly. In $C = -1$, all the paths pass through the origin in order to first switch the signs (change the surface) and then proceed to the solution space.



(b) The training paths for F_1. We observe that all the paths pass through the origin (change the surface) in order to switch the weights' signs in $C = -1$ surface.

Fig. 5. The training paths of different initializations. In the row plots (a, b, c) are for Boolean functions F_0, F_1, and F_5, respectively. The left column plots are for $C = 1$ and the right column plots are for $C = -1$. The shaded areas are the solution spaces.



(c) The training paths for F_5. There are two leaves in the solution space. The training paths converge to the upper leaf in $C = 1$ and to the lower leaf in $C = -1$, staying on the same surface. The two leaves are on different error surfaces.

Fig. 5. (Cont'd) The training paths of different initializations. In the row plots (a, b, c) are for Boolean functions F_0, F_1, and F_5, respectively. The left column plots are for $C = 1$ and the right column plots are for $C = -1$. The shaded areas are the solution spaces.

Table 2. The transition table.

We list the neighboring Boolean functions for each Boolean function according to Fig. 3. The Boolean functions that are in italics font have a two-bit difference in a transition; the functions that are not in italics have a one-bit difference in a transition. The arrows indicate the transition path that is shown in Fig. 4 (b).

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
F_1	F_3	F_3	F_{11}	F_5	F_4		F_5	F_{10}		F_{14}	F_{10}	F_4	F_5	F_{15}	F_{14}
F_4	F_5	F_{10}	F_7	F_{12}	F_1		F_3	F_{12}		F_{11}	F_3	F_8	F_{12}	F_{12}	F_{11}
F_8	F_0	F_0	F_1	F_0	F_{13}		F_{15}	F_0		F_2	F_{15}	F_{13}	F_{15}	F_{10}	F_7
F_2	*F_7*	*F_{11}*	F_2	*F_{13}*	F_7		*F_1*	*F_{14}*		F_8	*F_2*	F_{14}	*F_4*	*F_8*	F_{13}
F_3	*F_2*	*F_1*	*F_0*	*F_1*	*F_0*		*F_{11}*	*F_2*		*F_0*	*F_{14}*	*F_0*	*F_7*	*F_{11}*	*F_5*
F_{12}	*F_4*	*F_8*	*F_{15}*	*F_8*	*F_{15}*		*F_{13}*	*F_4*		*F_{15}*	*F_7*	*F_{15}*	*F_{14}*	*F_{13}*	*F_{10}*
F_5															*F_3*
F_{10}															*F_{12}*

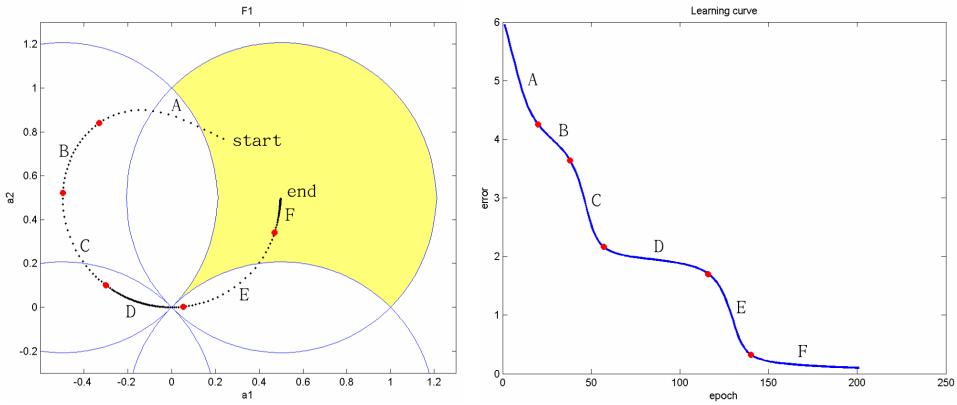
As shown in Fig. 5 (c), in the case of $C = 1$, the training paths are long for those initializations near the negative x_2 axis. The extreme cases are those initializations on the negative x_2 axis, which will require infinitely long training. The same situation exists for the case of $C = -1$ when the initializations are on the positive x_2 axis. These kinds of long paths also appear in the case shown in Fig. 5 (b), ($C = 1$).

Judging from Figs. 2 and 3, we can construct a transition table (see Table 2). In this table, we list all the possible transitions from a given Boolean area. All the transitions are in the neighborhood of this Boolean area, as shown in Fig. 3. In Fig. 3 we denote a specific area in the input space as a Boolean area, where this area is the solution space of its Boolean function. This table is useful for explaining and inferring the transitions in a training path. A transition example is plotted in this table for the case shown in Fig. 4 (b).

The BP algorithm will select a transition from its neighborhood with a reduced error in the same error surface before jumping to the other surface.

2.2 Learning Behaviors

Now, we will study learning behavior using an example presented previously (Fig. 4 (b)). From Fig. 6 (a), we observe that the period D in the learning curve (see Fig. 6 (b)) corresponding to the training path passes through the origin. The error surface near the origin is very complex. There is an extremely narrow and flat descending slope for the path, moving from the left leaf of F_3 to the right leaf of F_3. This causes slow learning. Fast learning periods will occur when the path passes through the Boolean area borders, such as periods C (F_{11} to F_3) and E (F_3 to F_1) in the learning curve. To our knowledge, this is the first explicit explanation of the slow and fast learning periods in the BP algorithm.



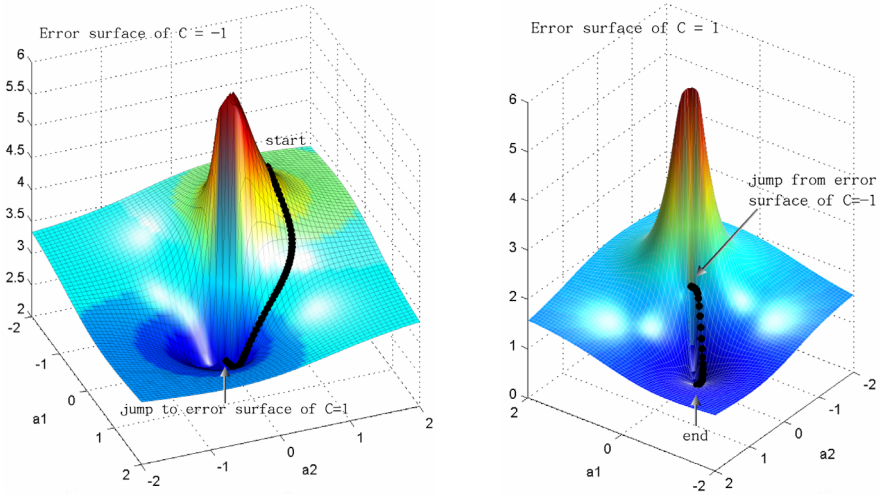
(a) (b)

Fig. 6. (a) The training path of initial weights w = [– 0.9, – 2.6, 2.2]. It is the same as in Fig. 4 (b); (b) The learning curve.

Fig. 7 shows a learning path of the decision line using the BP algorithm on the error surface with Eqs. (4) and (5). We see that this path descends and stays in one of the error surfaces marked C = – 1 in Fig. 7 (a), where it begins to evolve. This path does not jump to the lower error surface until it passes through the origin to switch the sign of w_3 (to change the sign of C from – 1 to 1 as shown in Fig. 7 (b)). Once the sign is reversed, the path evolves in the opposite error surface with C = + 1.

3. SIMULATION

To our knowledge, there is no feasible learning method that can resolve and accelerate learning in the region near the origin. The momentum method [2, 7, 8] can accelerate learning only in the slow slope regions that are far from the origin, we may apply the momentum method to only the region far from the origin.



(a) (b)

Fig. 7. The path jumps between the two sigmoid error surfaces (corresponding to Figs. 2 (4a, 4b)). The learning path starts from the surface $C = -1$ as shown in (a) and follows the descending gradient of that surface, it then reaches the origin $(a_1, a_2) = (0, 0)$, it changes to the surface $C = 1$ at the origin, it evolves to the minimum in the surface $C = 1$ as shown in (b).

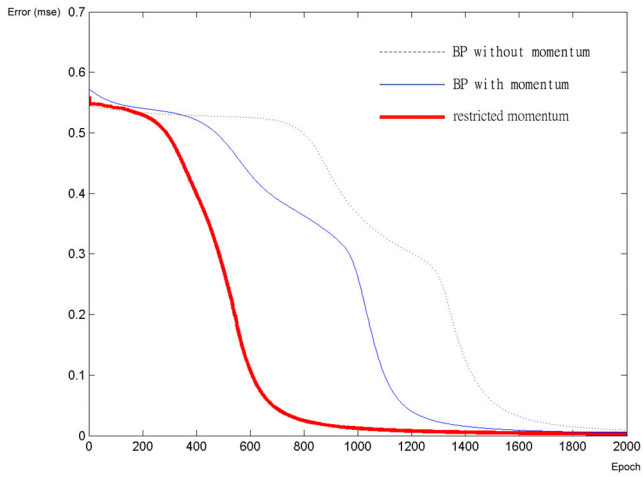


Fig. 8. The learning curves of BP (dashed line), BP with the momentum (thin line), and the restricted momentum method (thick line). The numbers on abscissa denote the training epochs. The ordinate denotes the mean square errors.

We performed a simulation with this restriction and compared the results obtained with BP method. We used a 2-3-1 feedforward network to solve the XOR problem. The momentum method was used only when the path was far from the origin with a distance larger than $R(\sqrt{\sum_i a_i^2} > R)$. We set $R = 0.8$ in the simulation. The initial weights of this network were set to small random numbers. The learning rate η was set to 0.7, and the momentum constant α was set to 0.3. The resulting learning curve is shown in Fig. 8. We

also plot the learning curves of the BP and momentum methods. As shown in this figure, learning with restricted momentum achieves better performance than conventional BP or BP with momentum.

REFERENCES

1. T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Transitions on Electronic Computers*, Vol. 14, 1965, pp. 326-334.
2. R. A. Jacobs, "Increased rates of convergences through learning rate adaptation," *Neural Networks*, Vol. 1, 1988, pp. 295-307.
3. Y. Lee, S. H. Oh, and M. W. Kim, "An analysis of premature saturation in back propagation learning," *Neural Network*, Vol. 6, 1993, pp. 719-728.
4. J. Li, A. N. Michel, and W. Porod, "Analysis and synthesis of a class of neural networks: linear systems operating on a closed hypercube," *IEEE Transactions on Circuits and Systems*, Vol. 36, 1989, pp. 1405-1422.
5. C. Y. Liou and H. T. Chen, "Self-relaxation for multilayer perceptron," *The 3rd Asian Fuzzy Systems Symposium* (*AFSS '98*), 1998, pp. 113-117.
6. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, Vol. 323, 1986, pp. 533-536.
7. F. M. Silva and L. B. Almeida, "Acceleration technique for the backpropagation algorithm," in *Proceedings of Neural Networks EURASIP Workshop*, L. B. Almeida and C. J. Wellekens (eds.), 1990, pp. 110-119.
8. R. L. Watrous, "Learning algorithms for connectionist networks: applied gradient methods of nonlinear optimization," in *FIRST IEEE International Conference on Neural Networks*, Vol. 2, 1987, pp. 619-627.
9. P. J. Werbos, "Backpropagation through time: what it does and how to do it," in *Proceedings of the IEEE*, Vol. 78, 1990, pp. 1550-1560.

Cheng-Yuan Liou (劉長遠) was born in Taiwan on November 14, 1951. He received the Ph.D. degree from Massachusetts Institute of Technology in 1985. He is Professor in the Department of Computer Science and Information Engineering. His interests include brain theory, neural networks, and computational mental process.

Jau-Chi Huang (黃昭綺) received the B.S. and M.S. degrees from Department of Computer Science and Information Engineering, National Taiwan University in Taipei, Taiwan in 2000 and 2002. She is currently working toward her Ph.D. degree in the same department. Her research interests are in neural network, artificial intelligence, and semantic coding.

Yen-Ting Kuo (郭彥廷) was born in Taichung, Taiwan, in 1977. He received the B.S. degree in Architecture from the National Cheng Kung University in 2001 and the M.S. degree in Computer Science and Information Engineering from the Taiwan University in 2003. His interests include art images, artificial intelligence, and neural networks.