Geometrical Perspective on Learning Behavior

Cheng-Yuan Liou[1], Jau-Chi Huang, Yen-Ting Kuo

Dept. of Computer Science and Information Engineering, National Taiwan University
Supported by National Science Council under Project NSC 91-2213-E-002-124
[1]correspondence , Email: cyliou@csie.ntu.edu.tw

Abstract. We construct a geometrical perspective to justify the slow learning period and fast learning period during training. We plot the error surfaces and the solution space on the input space for a single neuron with two inputs. We study various training paths on this space when we run the back-propagation (BP) learning algorithm [1]. We display the relation between the learning curve and the training path. We apply this study to correctly and efficiently operate the momentum method [2] to accelerate the training.

Keywords. multilayer network, back-propagation learning, neural network, momentum method

1 Introduction

The error surface, or energy surface, and the solution space for descending learning rules(BP) [1] have commonly been plotted in the weight space or in hypercube space [3,4]. Instead of this common approach, we draw the error surfaces and the solution space on the input space [5,6]. This will provide different viewpoint for the training paths. Consider a neuron with its weights [w_1, w_2, w_3] and two inputs {x_1, x_2}, plus a fixed input 1 for the threshold w_3 (see Fig. 1a). In Fig. 1b, L designates a decision line given by weights [w_1, w_2, w_3].

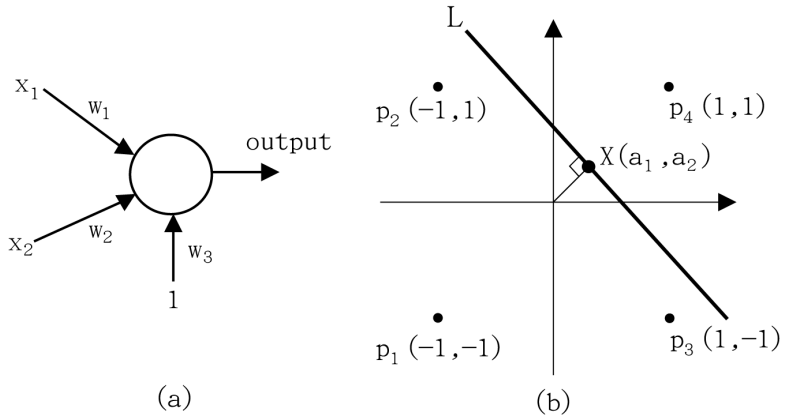


Fig. 1. (a) a single neuron diagram. (b) the decision line L.

The line L can be represented by a perpendicular point X at location (a_1, a_2), where a_1 and a_2 are obtained by solving the following equations:

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

Reformulate above equation, we obtain

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

We use such a decision point (a_1, a_2) to represent the decision line (or decision hyperplane). Accordingly, in this input space, each point (a_1, a_2) corresponds to two decision lines:

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

In the next section, we use the collection of such points (a_1, a_2) to represent error surfaces, solution space, and training paths. We 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 = 1~4.} as the inputs shown in Fig. 1b, to construct error surfaces, solution space, and training paths. The desired outputs of these four patterns have 2^4 =16 combinations. Each combination is a desired Boolean function. We list them as follows.

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 solution space

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

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

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

The variable $net_p^{(i)}$ of the sigmoid function σ (.) has two possible values. They are

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

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 the hard-limited activation function $\text{sgn}(net_p^{(i)})$.

We plot the two hard-limited error surfaces, C = ± 1, and the solution space where $E_X^{(1)} = 0$ or $E_X^{(2)} = 0$ in Fig. 2. Each point in the shaded area of the solution space represents an admissible decision line. Note that the minima of error in Fig. 2(1a), 2(2a), 2(3a), 2(4a), 2(5a) are not zero. There is no solution under these surfaces with *C*=-1 because the directions of their decision lines are wrong. The minima of error in Fig. 2(8a) and 2(8b) both are not zero so the solution space does not exist (see Fig. 2(8c)). This is called XOR problem. Other Boolean functions that are not shown in Fig. 2 can be obtained by changing the sign of patterns.

The solution space for these 14 Boolean functions (except XOR and XNOR) is summarized in Fig. 3. It is useful to examine the transition path of BP algorithm in such space which will be discussed in the successive section.

2.2 Training paths

We follow a sequence of decision points, or a training path, during BP training to show important learning behaviors. We will use the desired output F_1 as an example. The result is shown in Fig. 4 and it shows some interesting properties.

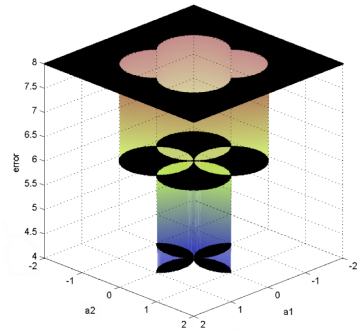
In Fig. 4a, the initial weights are [0.9, 2.6, 2.2], and the converged weights are [2.0, 2.0, -1.8]. We observe that the training path passes the origin ($w_3 = 0$), where the sign of w_3 switches. In Fig. 4b, the

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 the weights must be changed to achieve convergence. We observe that the training path first passes the x_2 axis, where the sign of w_1 changes. Then, it changes its direction to pass 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 the origin or any axis in order to change its signs. It will move to the solution directly (see Fig. 4c).

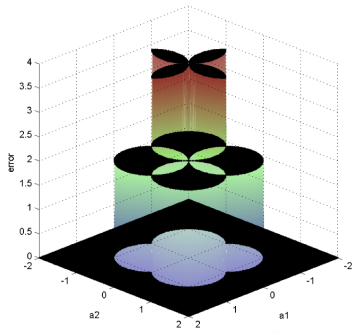
In many cases, as shown in Fig. 4a, 4b and Fig. 5a, 5b (C=-1), the path will pass either an axis or the origin, and the signs of the weights will switch. The path will jump to the other surface at the origin to reach the minimum. With this property, we may initially assign small magnitudes to the weights.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

As shown in Fig. 5c, in the case C=1, the training paths are long for 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 C=-1 when the initializations are on the positive x_2 axis. These kinds of long paths are also appeared in the case shown in Fig. 5b, (C=1).

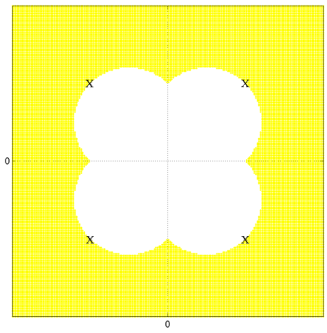
Judging from Fig. 2 and Fig. 3, we construct a transition table (Table 2). In this table we list all the possible transitions from a given Boolean area. All the transitions are in the neighborhood of this Boolean area as shown in Fig. 3. We denote a specific area in the input space as a Boolean area, where this area is the solution space of this 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. 4b. 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.



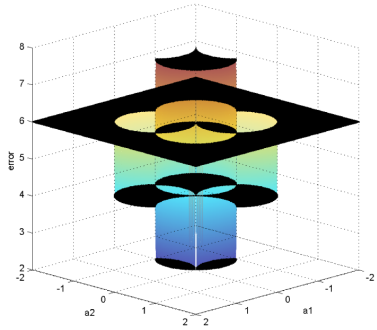
(1a)Error sruface. C=-1.



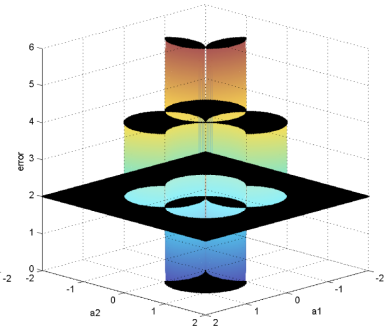
(1b)Error sruface. C=1.



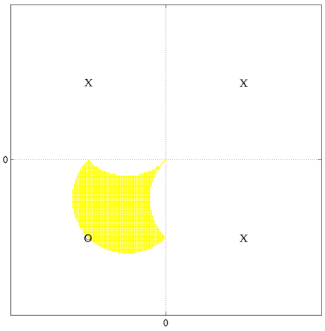
(1c)Solution space of F_0



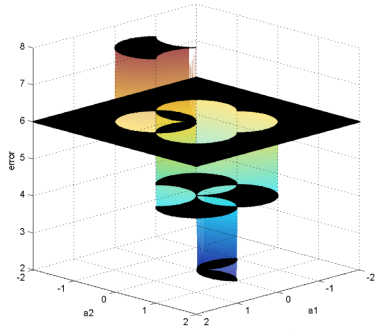
(2a)Error sruface. C=-1.



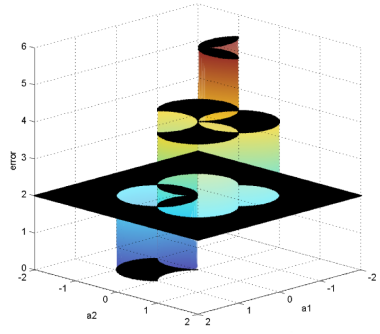
(2b)Error sruface. C=1.



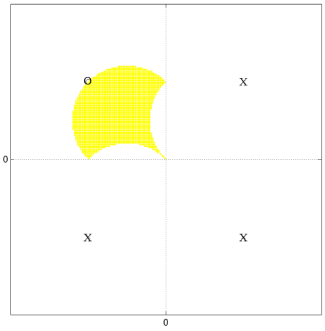
(2c)Solution space of F_8



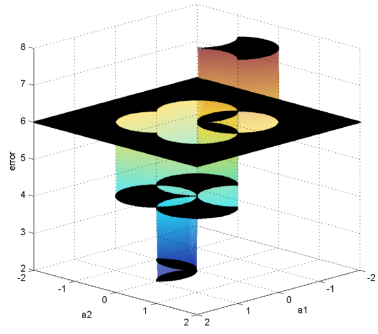
(3a)Error sruface. C=-1.



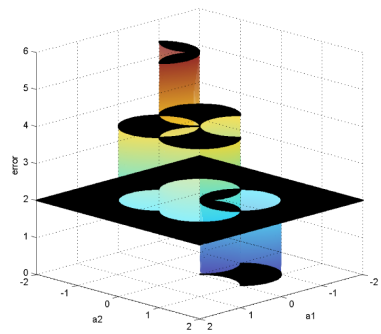
(3b)Error sruface. C=1.



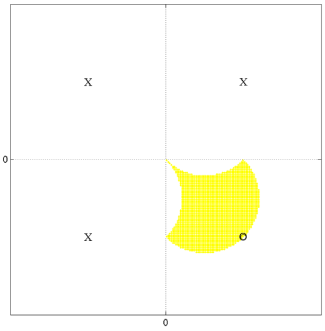
(3c)Solution space of F_4



(4a)Error sruface. C=-1.



(4b)Error sruface. C=1.



(4c)Solution space of F_2

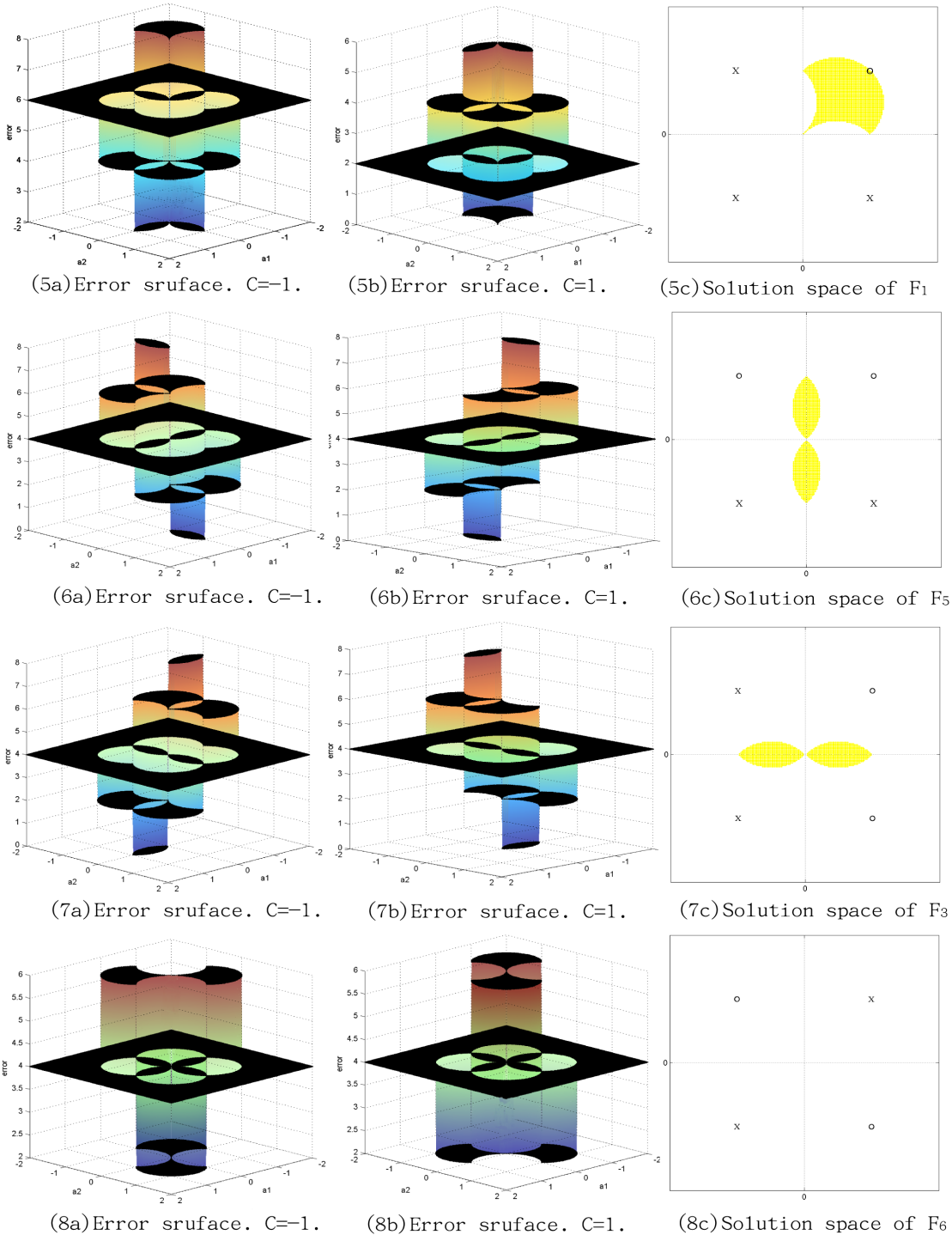


Fig. 2. The hard-limited error surfaces and the solution space for eight Boolean functions (F_0, F_8, F_4, F_2, F_1, F_5, F_3, F_6). Note that 'o' represents 1 and 'x' represents –1 in the desired response.

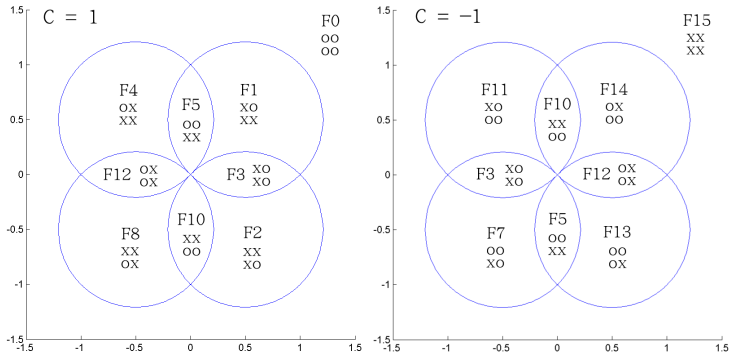


Fig. 3. The solution space of 14 Boolean functions. The 'o', 'x' marks are consistent with Fig. 2. Each closed area is a Boolean area.

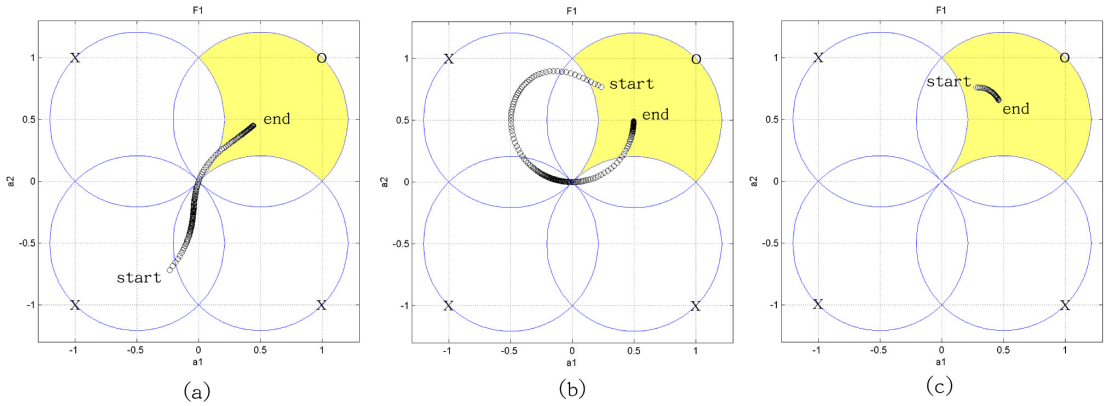


Fig. 4. (a, b, c) The training paths with different initial weights. (a) The initial weights are [0.9, 2.6, 2.2] and the converged weights are [2.0, 2.0, -1.8]. (b) The initial weights are [-0.9, -2.6, 2.2] and the converged weights are [1.9, 1.9, -1.9]. (c) The initial weights are [0.9, 2.6, -2.2] and the converged weights are [1.9, 1.9, -1.9].

Table 2. The transition table. The Boolean functions that are in italic font have a two-bit difference in a transition; the functions that are in normal font have a one-bit difference in a transition. The arrows indicate the transition path that is shown in Fig. 4b.

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}*

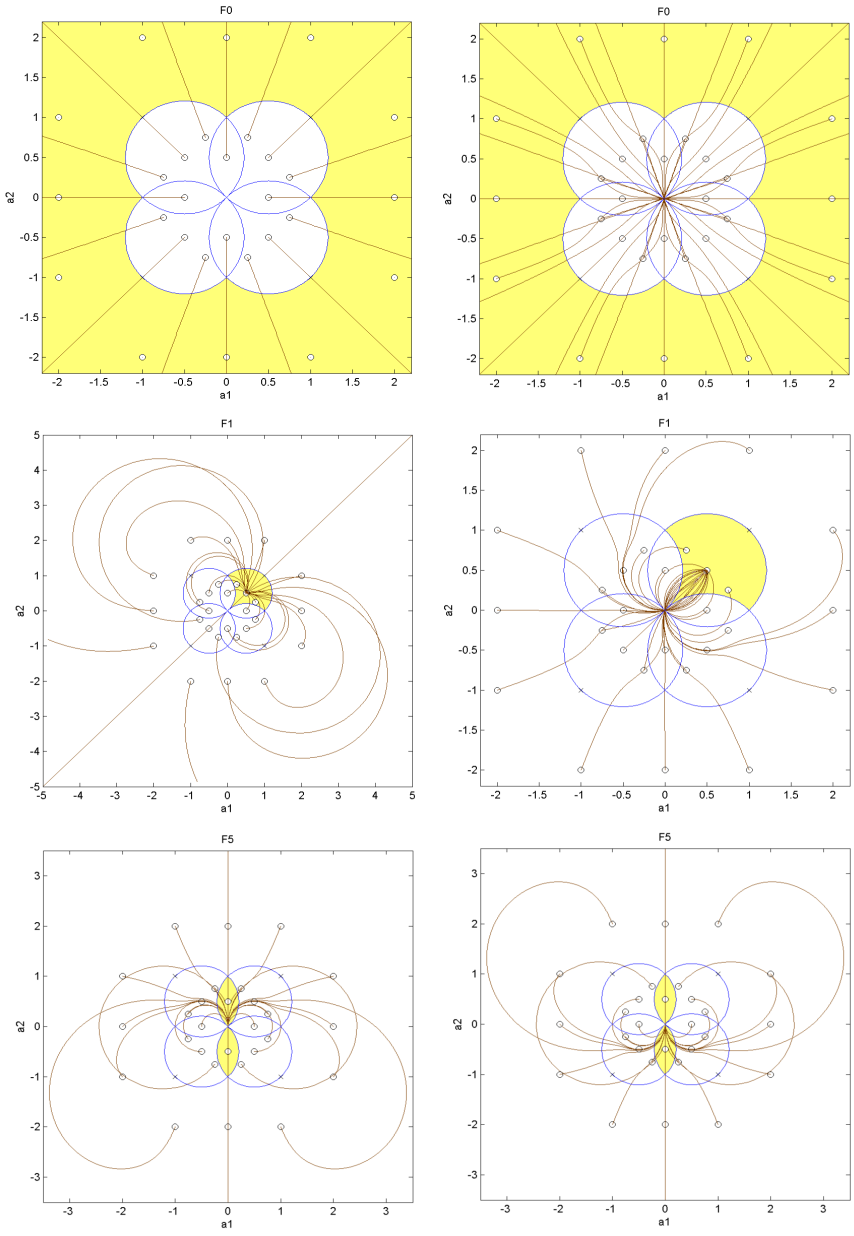


Fig. 5. The training paths of different initializations. (a, b, c) stands for Boolean function F_0, F_1 and F_5 respectively. The shaded area is the solution space. (a) The training paths for F_0. In C=1, all the paths proceed outwards to the solution space directly. In C=-1, all the paths pass the origin in order to switch the sign first (changing the surface) and then proceed to the solution space. (b) The training paths for F_1. We observe that all the paths pass the origin (changing the surface) in order to switch the weight's sign in C=-1. (c) The training paths for F_5. There are two leaves in the solution space. The training path converges 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.

2.2 Learning Behaviors

Now, we will study learning behaviors using an example presented previously (Fig. 4b). From Fig. 6a, we observe that the period D in the learning curve (see Fig. 6b) corresponding to the training path passes through the origin. The error surface near the origin is very complicated. 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 the Boolean area borders, such as periods C (F_{11} to F_3) and E (F_3 to F_1) in the learning curve. As far as we know, this is the first explicit explanation for the slow and fast learning period in BP algorithm.

Fig. 7 shows a learning path of the decision line using the BP algorithm on the error surface with Eq. 4, 5. We see that this path will follow down and keep in one of the error surface as marked by C=1 in Fig. 7a, where it starts the evolution. This path will not jump to the lower error surface until it passes the origin to switch the sign of w_3 (to change the sign of C from –1 to 1 as in Fig. 7b). Once the sign is reversed, the path will evolve in the opposite error surface with C=+1.

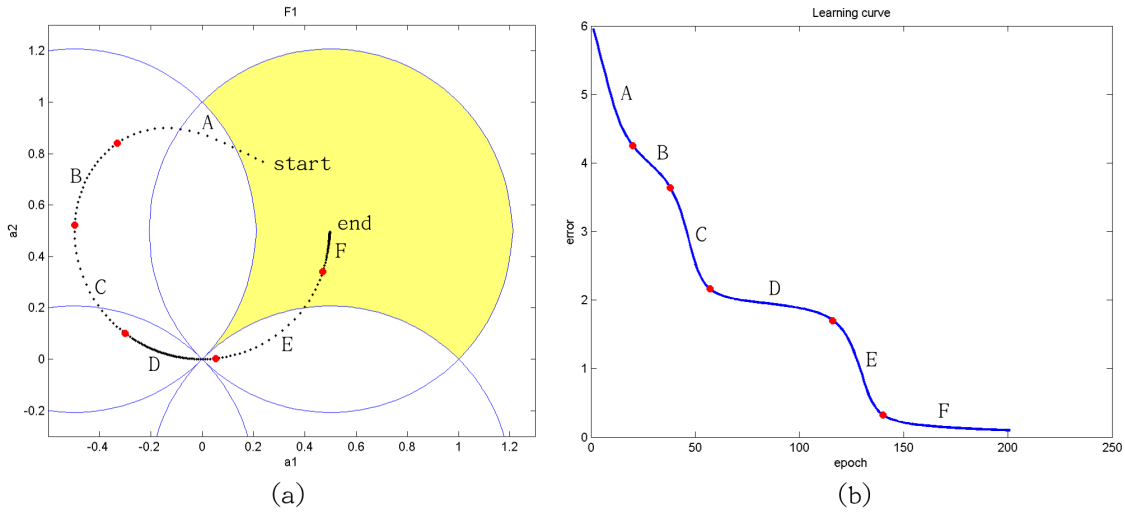


Fig. 6. (a) The training path of initial weights w = [-0.9, -2.6, 2.2]. It is the same as Fig. 4b. (b) The learning curve.

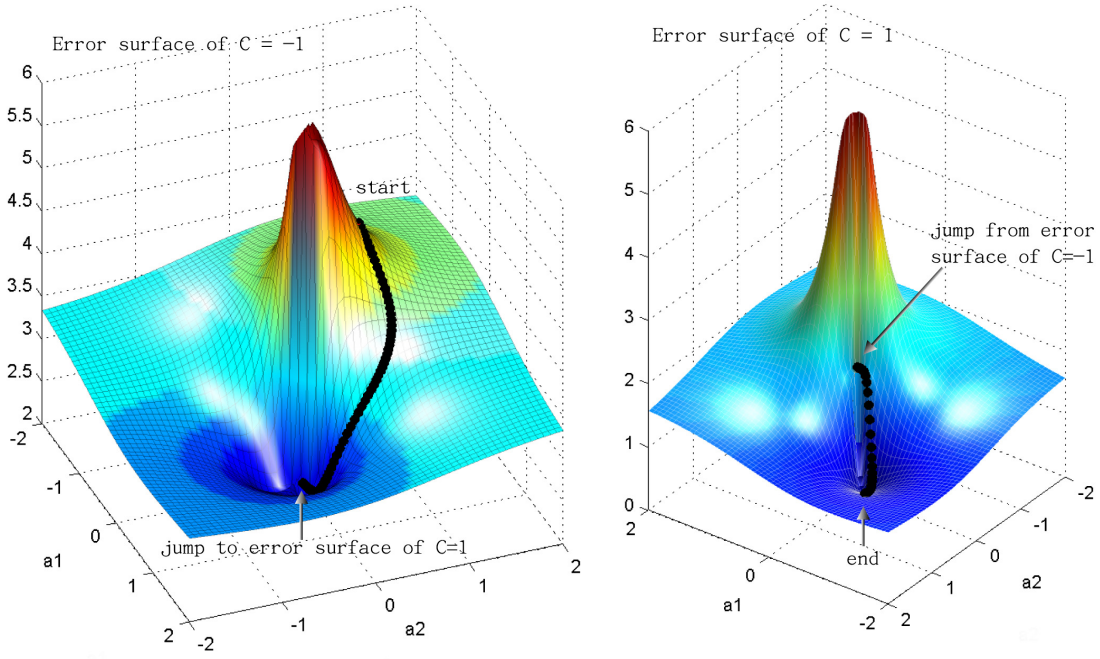


Fig. 7. The path jumps between the two sigmoid error surfaces (corresponding to Fig. 2(4a, 4b)). The learning path starts at (a) tracking the descending gradient, when it comes to the origin $(a_1,a_2)=(0,0)$, it changes the sign of C to 1 and the error surface becomes what is shown in (b). Then it still tracks the greatest gradient to the minimum error.

3 Simulation

To our knowledge, there is no feasible learning method which can resolve and accelerate the learning in the region close to the origin. Since the momentum method [2] can accelerate the learning in the slow slope region which is far from the origin. So we may restrict using the momentum method only in the region far from the origin.

We show a simulation with this restriction and compare it with both BP and BP with momentum methods. We use a 2-3-1 feed-forward network to solve the XOR problem. The momentum method is used only when the path is far from the origin with a distance larger than R ($\sqrt{\sum_i a_i^2} > R$).

Here we set R=0.8 in the simulation. The initial weights of this network are set to small random numbers. The learning rate η is set to 0.7 and the momentum constant α is set to 0.3. The result for the learning curve is shown in Fig. 8. We also plot the learning curves of the BP and momentum methods. As this figure shows, learning with restricted momentum achieves higher performance than conventional BP or BP with momentum.

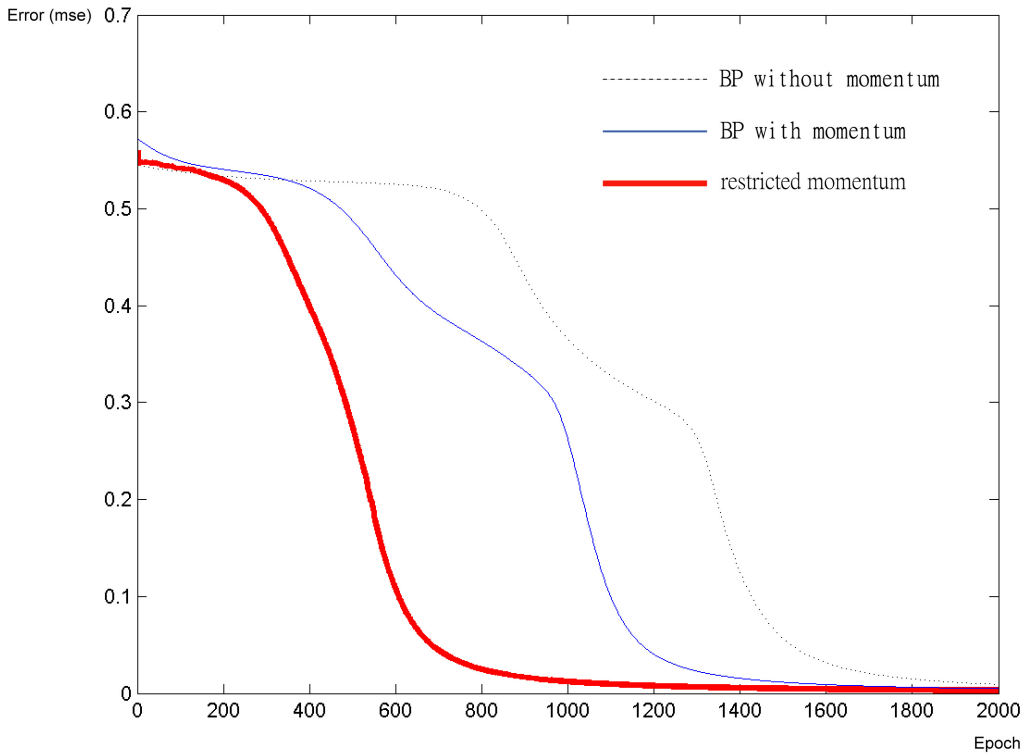


Fig. 8. The learning curves BP(dash line), BP with momentum(thin line), and the restricted momentum method(thick line). The numbers on abscissa denote the training epochs. The ordinate denotes the mean square errors.

References

1. D.E. Rumelhart, G.E. Hinton, and R.J. Williams, (1986), Learning Representations by Back-propagating Errors. Nature (London), Vol. 323, 533–536
2. R.A. Jacobs, (1988), Increased Rates of Convergences through Learning Rate Adaptation, Neural Networks, Vol.1, 295-307
3. T.M. Cover, (1965), Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition, IEEE Transitions on Electronic Computers, Vol.14, 326-334.
4. J. Li, A.N. Michel, W. Porod, (1989), Analysis and Synthesis of a Class of Neural Networks: Linear Systems Operating on a Closed Hypercube, IEEE Transactions on Circuits and Systems, Vol.36, No.11, 1405-1422.
5. C.Y. Liou and H.T. Chen, (1998), Self-Relaxation for Multilayer Perceptron, The Third Asian Fuzzy Systems Symposium, AFSS ’98, June 18-21, Masan, Korea, 113–117
6. C.Y. Liou, (2001), Lecture Notes on Neural Networks. National Taiwan University, 526 U1180. (http://red.csie.ntu.edu.tw/NN/index.html)