ANALYSIS OF SIMPLE NEURAL NETWORKS

Chedsada Chinrungrueng

Master's Report
Under the Supervision of Prof. Carlo H. Séquin

Department of Electrical Engineering & Computer Science
University of California, Berkeley
Berkeley, CA 94720

ABSTRACT

In a layered feed-forward network the *error surface* with respect to a desired goal function completely describes the potential of that network to carry out a particular task. Such an error surface can be obtained by plotting the maximum error or the sum of the squares of the errors, where the individual errors are the deviation of the actual output of the network from the desired goal function. For a network with W individual (synaptic) weights, these error surfaces are W-dimensional surfaces embedded in a $W+1$-dimensional space and defined over the W-dimensional domain determined by the ranges of the W weights.

We have studied in detail the error surfaces of a few small networks that we believe exhibit some of the typical characteristics also found in large networks. For the chosen networks we have determined the inherent symmetries of their associated error surfaces, the shape of the region around the origin, and the numbers and types of local minima. For each type of local minimum, we examined its multiplicity, the fractional coverage of the total domain space by its collection zone, and the approximate shape of the valleys leading into it. This study of small three-layered networks performing the Exclusive-OR function of two or three inputs provides some insight into the general structures of the error surfaces and thus the capabilities of more complicated networks.

December 20, 1988

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
20 DEC 1988		**00-00-1988 to 00-00-1988**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Analysis of Simple Neural Networks	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Berkeley,Department of Electrical Engineering and Computer Sciences,Berkeley,CA,94720**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In a layered feet-forward network the error surface with respect to a desired goal function completely describes the potential of that network to carry out a particular task. Such an error surface can be obtained by plotting the maximum error or the sum of the squares of the errors, where the individual errors are the deviation of the actual output of the network from the desired goal function. For a network with W individual (synaptic) weights, these error surfaces are W-dimensional surfaces embedded in a W + 1-dimensional space and defined over the W-dimensional domain determined by the ranges of the W weights. We have studied in detail the error surfaces of a few small networks that we believe exhibit some of the typical characteristics also found in large networks. For the chosen networks we have determined the inherent symmetries of their associated error surfaces, the shape of the region around the origin, and the numbers and types of local minima. For each type of local minimum, we examined its multiplicity, the fractional coverage of the total domain space by its collection zone, and the approximate shape of the valleys leading into it. This study of small three-layered networks performing the Exclusive-OR function of two or three inputs provides some insight into the general structures of the error surfaces and thus the capabilities of more complicated networks.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**66**	

Table of Contents

Acknowledgements

This work was supported in part by the Joint Services Electronics Program contract (number F49620-87-C0041) and by a Gift from NEC Corporation.

I would also like to thank my research advisor, Prof. Carlo H. Séquin, for his continuous encouragement and guidance. I have learned a great deal from his teaching, knowledge, and criticism.

ANALYSIS OF SIMPLE NEURAL NETWORKS

Chedsada Chinrungrueng

Master's Report
Under the Supervision of Prof. Carlo H. Séquin

Department of Electrical Engineering & Computer Science
University of California, Berkeley
Berkeley, CA 94720

1. MOTIVATION

A neural network is a system that achieves computational power via a massively parallel network composed of many simple computational elements arranged in patterns similar to those found in some biological neural systems. Computational elements are connected by links that can vary in strengths called *weights*. The patterns of connections, which are called *architectures*, have a great effect on the computational power of the network. Networks with different architectures have different computation capabilities. Although, we have some knowledge about such networks, their real potential will remain a conjecture until their characteristics are better understood.

One example of an architecture is the layered feed-forward network. This class of networks has recently met with great interest because of its ability to represent complex mappings. Moreover, it is claimed that this kind of network is able to capture regularities in the specified mapping [1]. Specifically, if there is a regularity in the mapping, these networks can exploit this regularity to generate the correct output patterns for input patterns they have never encountered before; a task called *generalization*. The capability of a layered feed-forward network to represent a particular mapping is completely described by the corresponding *error surface*. The error surface, usually high dimensional, measures the deviation between the desired mapping and the actual mapping performed by the network for all possible combinations of link strengths, i.e., values of its weights. Understanding the structure of the error surface will help us answer many questions about the network's properties.

The goal of this report is to increase understanding of layered feed-forward networks by studying their error surfaces. We also wish to present new visualization techniques for high dimensional error surfaces. We have chosen to concentrate on problems concerning error surfaces associated with the networks performing the exclusive-or (XOR) function and the parity function of three inputs. These are among the simplest problems that cannot be solved with a simple two-layer network. For these cases, the error surfaces are mapped out and presented in a form that are intuitively understandable. We believe that these simple studies provide insight into the structure of the error surfaces of more complicated neural networks.

2. LAYERED FEED-FORWARD NETWORKS

A layered feed-forward network is a type of neural network that consists of layers of simple computational elements. These computational elements, referred to as *neurons*, are connected by links with variable weights.

In a layered feed-forward network only neurons belonging to different layers are connected by links. The first, or bottom, layer is an input layer. The neurons in the input layer are not actual computational elements. Their task is simply to provide an impedance match to the input signals. The last, or top, layer is an output layer. Every observable output variable is associated with one output neuron. Between the input and the output layers there can be any number of hidden layers. Each neuron in the hidden layers must receive its input signal from the neurons in the layers lower than its own and must send the output signal to neurons in layers higher than its own. When given an input vector, the network computes its output by a forward pass. In this event all neurons in each layer receive vector signals from lower layers and compute outputs for subsequent layers.

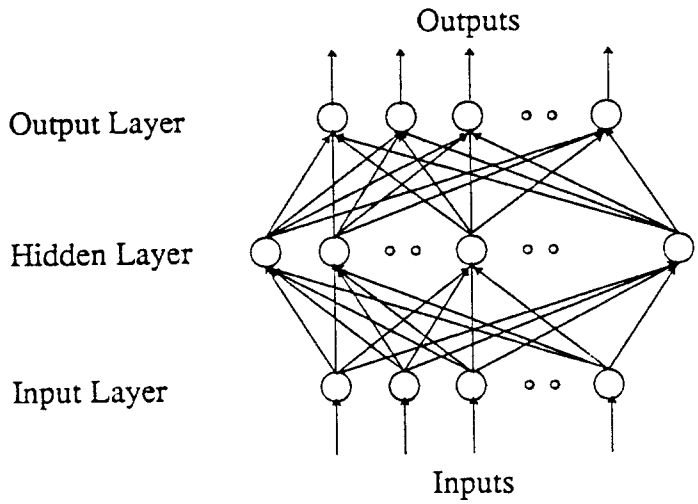


Fig. 1 A Network with Complete Connection between Adjacent Layers

Computational elements, or neurons, are usually non-linear analog processors capable of performing simple tasks, such as summing, thresholding, scaling. A typical model of a neuron is a processor that forms the weighted sum of all its input signals, adds a bias term to the sum, and then passes the result through a switching function as shown in Fig 2. The value of the function becomes the output of the neuron.

The particular networks considered in this report belongs to a class of three-layer networks with complete connectivity between the adjacent layers. A sample of this class of network is illustrated in Figure 3. The network is composed of three layers of neurons : an input layer (L1), a hidden layer (L2), and an output layer (L3). Neurons in one layer are connected only to neurons in the subsequent layer. That is, the neurons in L1 are connected only to neurons in L2 but are not connected directly to neurons in L3.

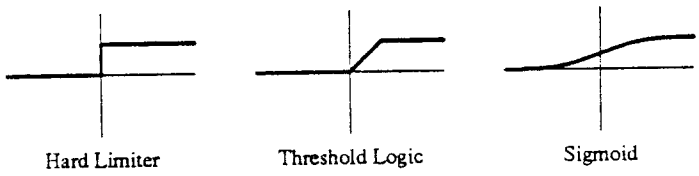


Fig. 2 Various Types of Switching Functions

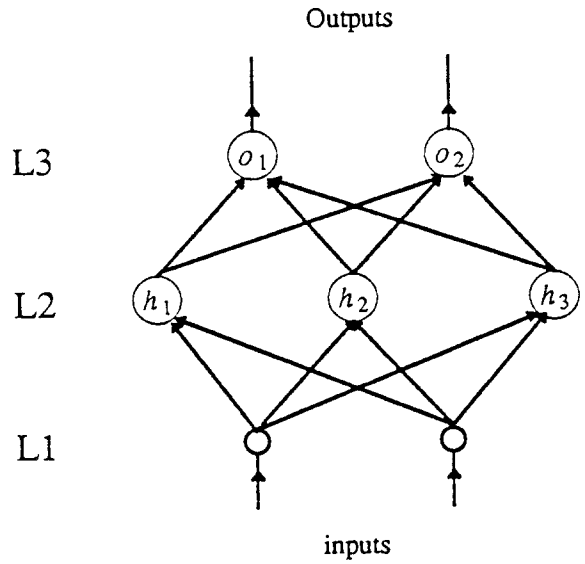


Fig. 3 A Sample of the Class of Networks Studied in this Report.

To see how the neurons in this class of networks compute their output, let us consider neuron o and neurons $h_1, h_2, \ldots, h_n$ as shown in Fig. 4.a. In this figure, neurons $h_1, h_2, \ldots, h_n$ provide input signals to neuron o through connections with strengths of $w_1, w_2, \ldots, w_n$. When the neuron o receives inputs $i_1, i_2, \ldots, i_n$, which are the outputs of the neurons $h_1, h_2, \ldots$, and h_n respectively, it forms the weighted sum of all the incoming signals by multipling each input signal with its corresponding weight and adding the products. Also, a bias term w_{n+1} is added to the weighted sum, and the new sum is passed through the sigmoid function such as that demonstrated in Fig. 4.b. This sigmoid function has been chosen to be symmetric with respect to the origin in order not to conceal any symmetry inherent in the topology of the network or in the specified mapping. Also, we normalize the value of the function so that the output of a neuron is in the range of [-1,1].

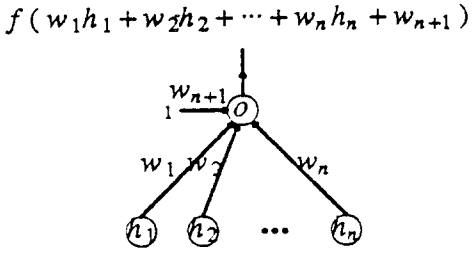


Fig. 4.a Input-Output Relation of a Neuron

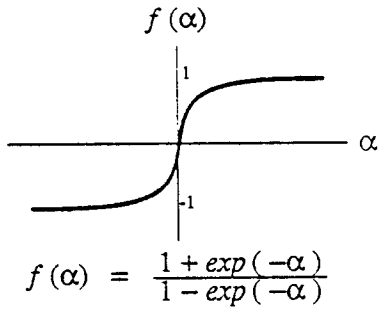


Fig. 4.b Sigmoid Function

3. MATHEMATICAL DEFINITIONS

3.1. Input-Output Relation of a Layered Feed-Forward Network

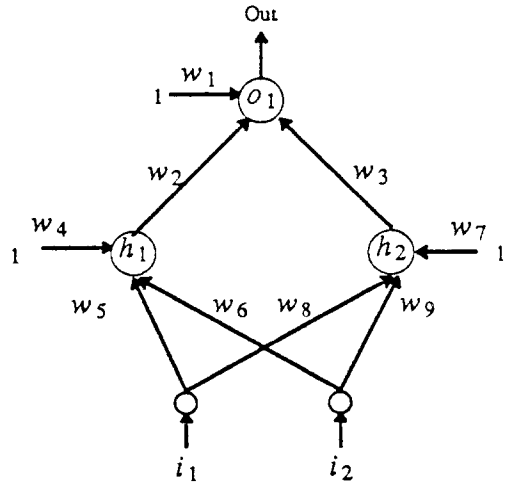


Fig. 5 A Network with Two Input Neurons and Two Hidden Neurons

When an input is presented to a layered feed-forward network, the signals are propagated from the input layer through hidden layers to the output layer. To find the input-output relation of a layered feed-forward network, let us consider the example shown in Figure 5. This network consists of two input neurons, two hidden neurons and one output neuron. In all but the input neurons, the sigmoid function illustrated in Figure 4.b is used as a switching function. These five neurons in the network are connected by nine adjustable weights labeled according to Figure 5. The values of these weights can be represented by an ordered nine-tuple of real numbers (w_1, w_2 , ... , w_8, w_9) and be conveniently referred to as a nine-dimensional point or a vector with nine components. Let us denote an input vector (i_1, i_2) by i, and a vector (w_1, w_2 , ... , w_8, w_9) representing a set of the weights of the network by **w**. The output values of the hidden neurons, h_1 and h_2 can be computed by the following equations :

$$h_1(\mathbf{w}, \mathbf{i}) = swf(w_4 + w_5 i_1 + w_6 i_2), \tag{1}$$

$$h_2(\mathbf{w}, \mathbf{i}) = swf(w_7 + w_8 i_1 + w_9 i_2), \tag{2}$$

where swf is the sigmoid switching function. The output neuron uses h_1 and h_2 to compute its output (o_1), which is the output of the network, according to the following equation:

$$o_1(\mathbf{w}, \mathbf{i}) = swf(w_7 + w_8 i_1 + w_9 i_2). \tag{3}$$

Equations 1, 2 and 3 define the input-output relation of the network for a given set of weights w. This overall input-output relation of the network can be summarized into one equation:

$$\mathbf{o} = O(\mathbf{w}, \mathbf{i}). \tag{4}$$

For this particular network, the input-output relation is a function from R^2 to R with w as a vector-valued parameter.

For a network with m input neurons and n output neurons, an input vector i becomes a vector with m components and an output vector o becomes a vector with n components. We call the set containing all possible input vectors an *input space* and the set containing all possible output vectors an *output space*. In this case, the input space is a subset of R^m and the output space is a subset of R^n. Furthermore, if we assume that the network consists of k weights, then w becomes a vector with k components. We call the set containing all possible w a *weight space*. When we relate equation (4) to this case, the input-output relation of the network will be a function from an input space to an output space with w, a vector in a weight space, as a parameter.

3.2. Mapping Representability of a Network

As the previous section mentions, the input-output relation of a layered feed-forward network is a function whose mapping rule depends on the connection weights of the network. That is, by altering the weights, the network can be used to implement different mappings. Suppose we want the network to perform the *goal–mapping* G, which is defined as $\{ (i_1, g_1), (i_2, g_2), \ldots, (i_g, g_g) \}$, where (i_p, g_p) are respective the input-output pair. If we restrict the input space of the network to the domain of G, denoted as $D(G)$, the restricted input-output relation of the network becomes a mapping with the domain equal to $D(G)$ and with the mapping rule $O(w,)$. We call the mapping produced by the network for this restricted input set an *actual* mapping and we denote it as $A(w)$. By comparing $A(w)$ to mapping G, we can see how well the network performs a specified task when it is assigned a set of weights w.

As an alternative to describing mapping G by specifying all input-output pairs, let us describe mapping G by specifying its domain $D(G)$ and its range $R(G)$. We order the elements in $R(G)$ so that the p-th element of $R(G)$ is an image of the p-th element of $D(G)$ under mapping G, for $p \in 1, 2, \cdots, g$. According to this method, mapping G can be described as a mapping with

$$\text{domain } D(G) = \{ i_1, i_2, \cdots, i_g \}, \tag{5}$$

and

$$\text{range } R(G) = \{ g_1, g_2, \cdots, g_g \}. \tag{6}$$

As $A(w)$ and G have the same domain, by keeping the elements of the domain of $A(w)$ in the same order as those of $D(G)$, we can describe mapping $A(w)$ as a mapping with

$$\text{domain } D(A(w)) = \{ i_1, i_2, \cdots, i_g \}, \tag{7}$$

and

$$\text{range } R(A(w)) = \{ o_1, o_2, \cdots, o_g \}. \tag{8}$$

where $o_p = O(w, i_p)$. Since the domains of G and $A(w)$ are the same, we can define a *difference* mapping, denoted as $(G - A(w))$, as follows :

$$(G - A(w))(i_p) = G(i_p) - A(w, i_p); \tag{9}$$

for every i_p in the domain of G and $A(w)$. This new mapping has the same domain as that of G or of $A(w)$ and the range which is described as :

$$\text{range of } (G - A(w)) = \{ (g_1 - o_1), (g_2 - o_2), \cdots, (g_g - o_g) \}. \tag{10}$$

By examining the magnitude of the elements in the range of ($G - A(\mathbf{w})$), we can find out how well the network with weights w can represent mapping G. For example, if all the elements in the range of mapping ($G - A(\mathbf{w})$) are zero, this function will map every element in its domain to 0. It means that $A(\mathbf{w})$ and G are the same, and in this case the network with weights w can exactly represent mapping G. On the contrary, if the magnitudes of most elements in the range of mapping ($G - A(\mathbf{w})$) are large, $A(\mathbf{w})$ and G will be are quite different. This implies that the performance of the network with weights w is poor in representing mapping G.

3.3. Error Function and Error Surface

Rather than using the range of mapping ($G - A(\mathbf{w})$) to reflect the performance of the previously discussed network, it is preferable to use only one scalar value to measure the performance of this network for representing mapping G. Therefore, we will associate the range of mapping ($G - A(\mathbf{w})$) with a given norm and then use this norm as a performance measure. Suppose the network has n output neurons and mapping G consists of g ordered pairs. This implies that there are also g ordered pairs in mapping ($G - A(\mathbf{w})$). According to the mapping description mentioned above, the range of ($G - A(\mathbf{w})$) must have g elements and may be described by equation (10). Reconsidering this equation, let us substitute[1] ($g_p^1, g_p^2, \cdots, g_p^n$) for $\mathbf{g}_p$ and ($o_p^1, o_p^2, \cdots, o_p^n$) for $\mathbf{o}_p$. The range of ($G - A(\mathbf{w})$) then becomes

$$\{ ((g_1^1 - o_1^1), (g_1^2 - o_1^2), \ldots, (g_1^n - o_1^n)), \quad ((g_2^1 - o_2^1), (g_2^2 - o_2^2), \ldots, (g_2^n - o_2^n)),$$
$$\cdots, \quad ((g_g^1 - o_g^1), (g_g^2 - o_g^2), \ldots, (g_g^n - o_g^n)) \}.$$

Since we are not concerned with each individual vector, we can rewrite the set of vectors specified above as a single vector:

$$((g_1^1 - o_1^1), (g_1^2 - o_1^2), \ldots, (g_1^n - o_1^n), (g_2^1 - o_2^1), (g_2^2 - o_2^2), \ldots, (g_2^n - o_2^n),$$
$$\cdots, (g_g^1 - o_g^1), (g_g^2 - o_g^2), \ldots, (g_g^n - o_g^n)).$$

We call this ordered gn-tuple a vector representing the range of ($G - A(\mathbf{w})$). The magnitudes of the elements in the range of ($G - A(\mathbf{w})$) reflect the difference between mapping G and mapping $A(\mathbf{w})$. They are summarized in the *norm* of the vector representing the range ($G - A(\mathbf{w})$), which can thus be viewed, intuitively, to represent the *error* between mapping G and mapping $A(\mathbf{w})$. We thus call the function that maps w, a point in the weight space, to this error value the *error function*, with respect to the goal mapping.

The definition of an error function depends on the definition of the vector norm. With a Euclidean norm,[2] the error function is expressed as

$$\text{EucErr}(\mathbf{w}) = \left(\sum_{i=1}^{n} \sum_{j=1}^{g} (g_j^i - O^i(\mathbf{w}, \mathbf{i}_j))^2 \right)^{\frac{1}{2}} \tag{11}$$

However, for ease of computation we will employ the square of the Euclidean norm in our study cases. Therefore, our error function is defined as

$$\text{TsqErr}(\mathbf{w}) = \left(\sum_{i=1}^{n} \sum_{j=1}^{g} (g^{i_j} - O^i(\mathbf{w}, \mathbf{i}_j))^2 \right) \tag{12}$$

---

[1] We use a superscript to indicate output neuron number.

[2] For vector x specified as ($x_1, x_2, \cdots, x_n$), the Euclidean norm is defined as $(x_1^2 + x_2^2 + \cdots + x_n^2)^{\frac{1}{2}}$.

If we are interested in the maximum error made for any of the elements in the domain of G, a norm defined as :

$$l^{\infty}(\mathrm{x}) = \max(|x_1|, |x_2|, \cdots, |x_n|). \tag{13}$$

should be used. This gives the error function of the form :

$$\mathrm{MaxErr}(\mathrm{w}) = \max | g_j^i - O^i(\mathrm{w}, \mathrm{i}_j) |, \tag{14}$$

for $i \in \{ 1, 2, \ldots, n \}$, and $j \in \{ 1, 2, \ldots, g \}$.

For a given network and a given problem statement, an error function that maps a set of weights to an error value fully describes how well the network performs for all possible values of its weights. Since the error function that we use in this report is continuous, we can use this function to define a surface called an *error surface*.[3] By studying this surface, we can know how well the network can be used to represent a designated mapping. Unfortunately, these error surfaces are of very high dimensions and thus cannot be easily perceived. We shall now discuss some visualization techniques to study such hypersurfaces.

[3] A surface of dimension n, or n-surface, is a set in R^{n+1} that contains all the points of the form $(\mathrm{c}, f(\mathrm{c}))$, where f is a smooth function from $\mathrm{R}^n \rightarrow \mathrm{R}$ and c belongs to the domain of f.

4. MODELING ERROR SURFACES

For a layered feed-forward network with k variable weights, an error surface is an k-dimensional manifold embedded in an $k+1$-dimensional space and defined over an k-dimensional space given by the weight space. This error surface completely describes the potential of the network for its desired goal mapping. Therefore, understanding this surface will help us answer all the questions related to the capability of the associated network in performing a designated mapping. For a very simple network, such as the Perceptron with two inputs, error surfaces are surfaces of two or three dimensions, and thus can be displayed with regular graphical techniques. However, for a more complicated system, error surfaces are usually of very high dimension where a simple visualization is impossible. Accordingly, for high-dimensional surfaces we need to consider ways other than those of modeling the surfaces. One alternative approach for portraying such hyper-surfaces is to present them in conceptual forms that reflect *significant* structures of those surfaces. Exactly what *significant* means depends on the particular questions that concern us.

In this report, we have chosen to concentrate on problems concerning error surfaces associated with networks performing Boolean functions. Since the sigmoid function employed as a switching in our neurons has its output value defined over [-1, 1], we use the value of 1 to represent the *on* state and the value of -1 to represent the *off* state. This representation also preserves symmetries inherent in the mapping statement. The importance of these symmetries will be fully discussed later.

4.1. Error Surfaces of a Perceptron with Two Inputs

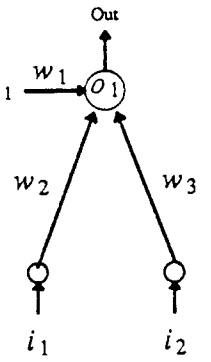


Fig. 6 A Perceptron with Two Inputs

For a layered feed-forward network with two or three variable weights, the error surface has two or three dimensions, and thus can be actually modeled. This is illustrated by the error surfaces associated with the Perceptron with two inputs, shown in Figure 6. This system has three variable weights; therefore, the error surfaces affiliated with this system will be 3-dimensional surfaces. One way to model these surfaces is by displaying a family of contour plots, which is demonstrated in Figure 7. From Figure 7.3, which illustrates the error surface of the Perceptron performing the XOR mapping, we see that no local minimum exists deep enough to constitute a good solution. Therefore, the single Perceptron cannot perform the XOR mapping.

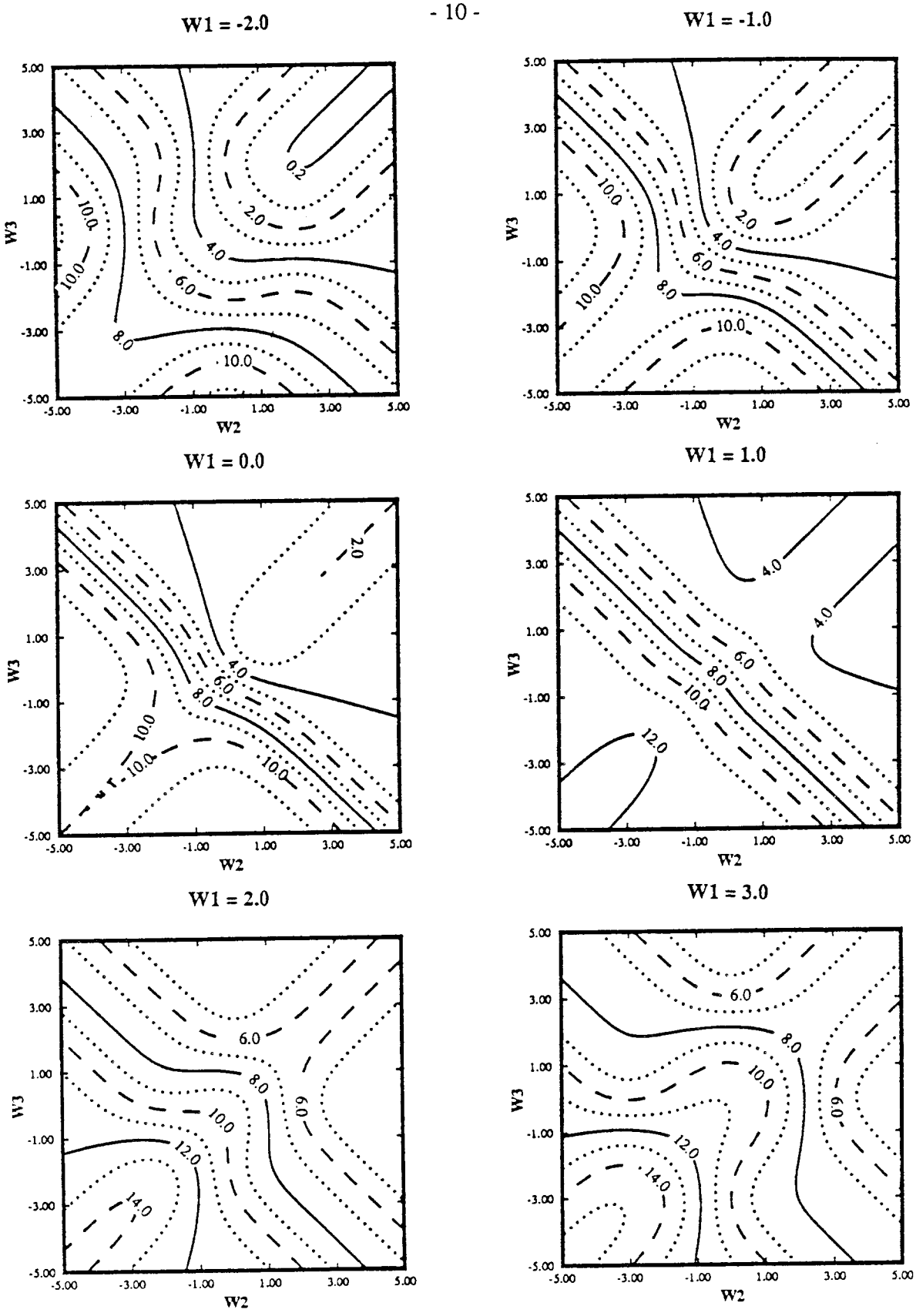


Fig. 7.1 Error Surface Associated with Perceptron Performing AND Function

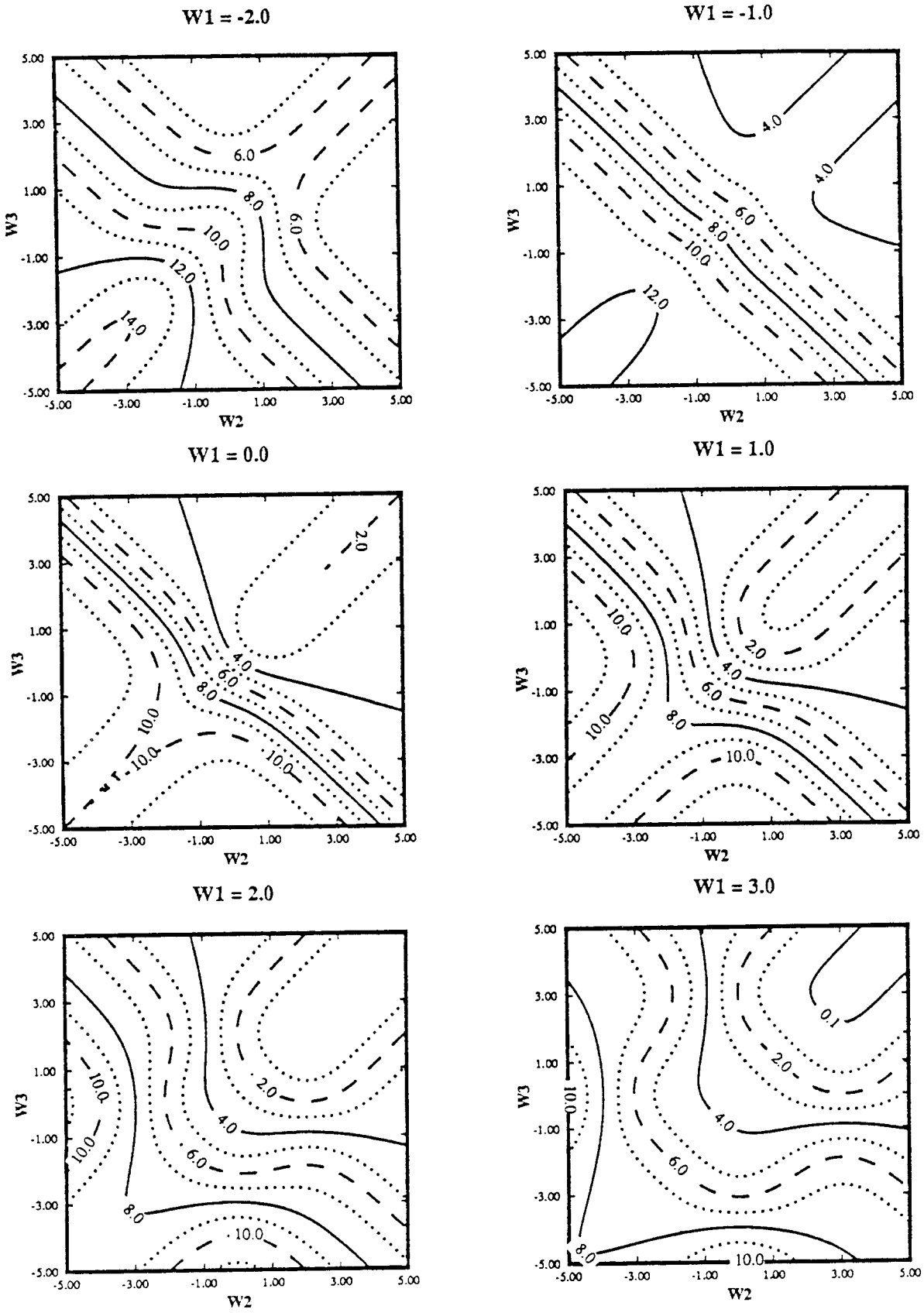


Fig. 7.2 Error Surface Associated with Perceptron Performing OR Function

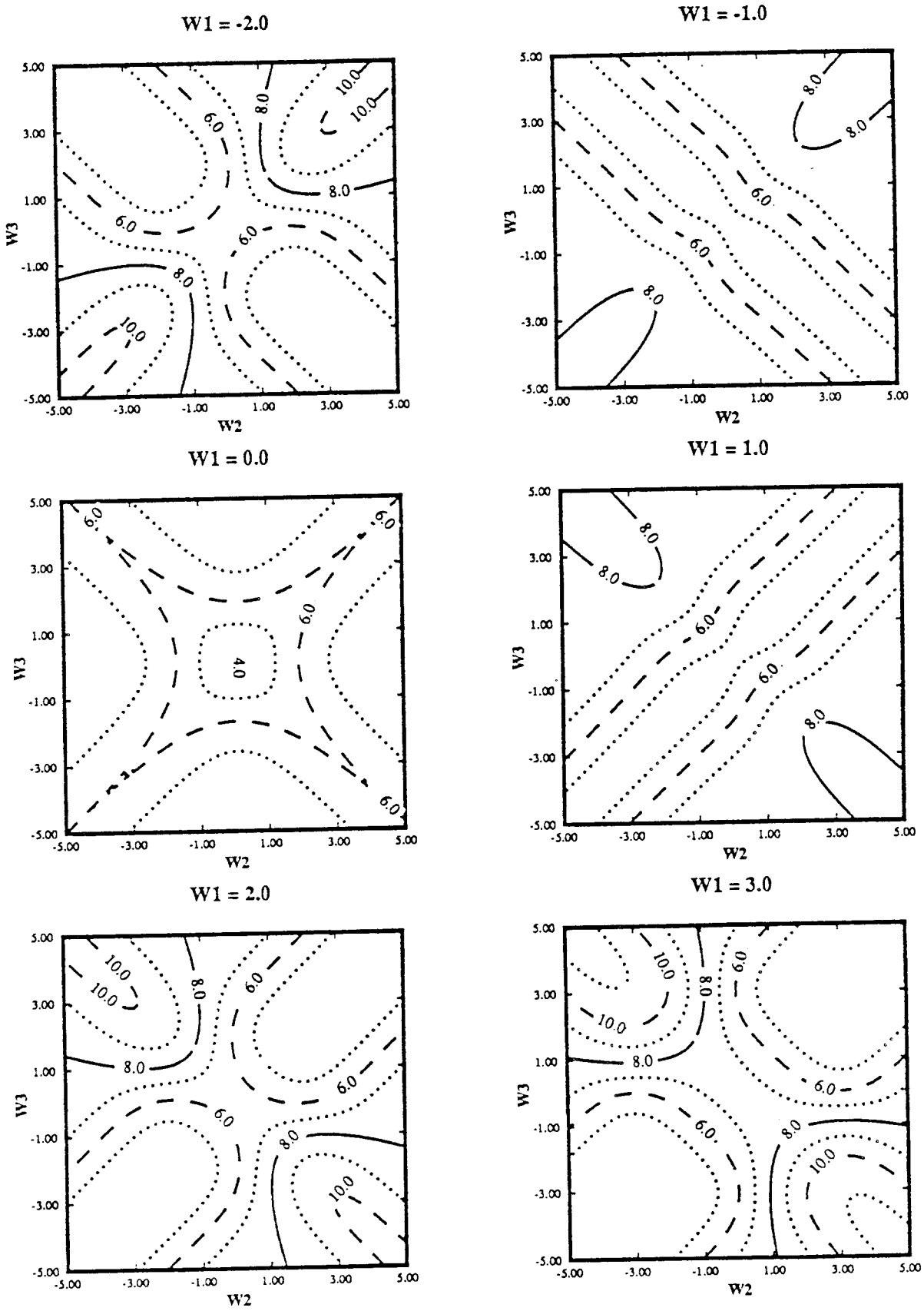


Fig. 7.3 Error Surface Associated with Perceptron Performing the XOR Function

4.2. High-Dimensional Error Surfaces

To gain some general insight into the structures of high-dimensional error surfaces, we shall study the error surfaces associated with the following problems :

- the network in Figure 8.1 with 2 hidden neurons performing the XOR function,
- the network in Figure 8.2 with 3 hidden neurons performing the XOR function,
- the network in Figure 8.3 with 4 hidden neurons performing the XOR function,
- the network in Figure 8.4 with 3 hidden neurons performing the 3-input odd-parity function,

In these study cases, we shall map out the error surfaces and then portray the derived information in terms of tables and graphs. We are interested in the potential of these networks to represent the designated mappings and in the efficiency of the networks in learning to perform such a mapping. Our *conceptual* models will concentrate on the following information :

- the nature of the error surface at the origin,
- the number of different types of minima and the multiplicity of each,
- the approximate shapes of the valleys affiliated with the various minima, and
- the fractional coverage of the total domain space by the collection zone of each minimum.[1]

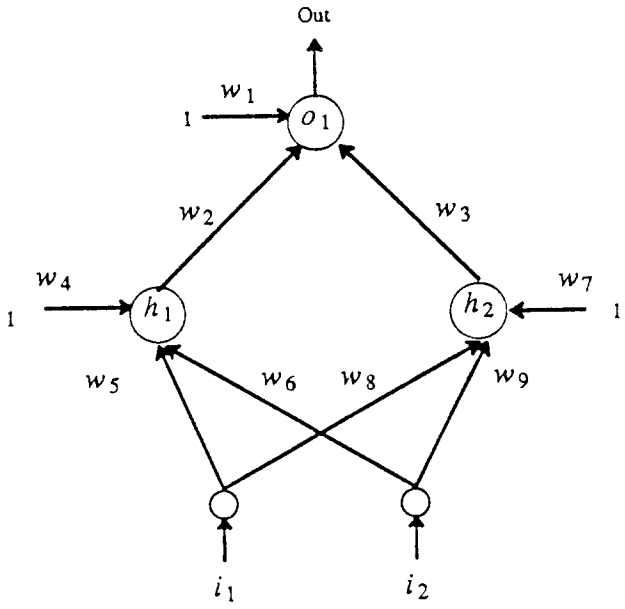


Fig. 8.1 A Network with Two Input Neurons and Two Hidden Neurons

[1] The collection zone of a given minimum is the region for which a steepest descent ends up in the given minimum.

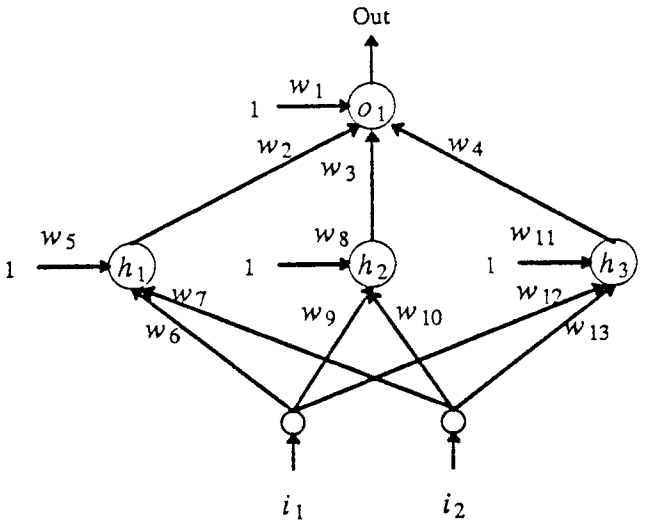


Fig. 8.2 A Network with Two Input Neurons and Three Hidden Neurons

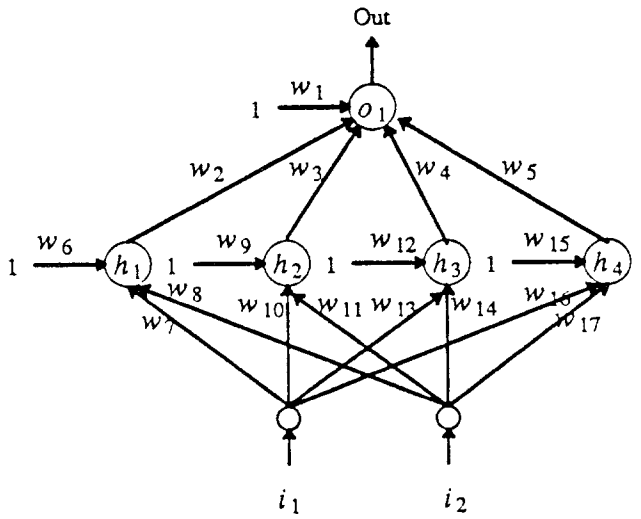


Fig. 8.3 A Network with Two Input Neurons and Four Hidden Neurons

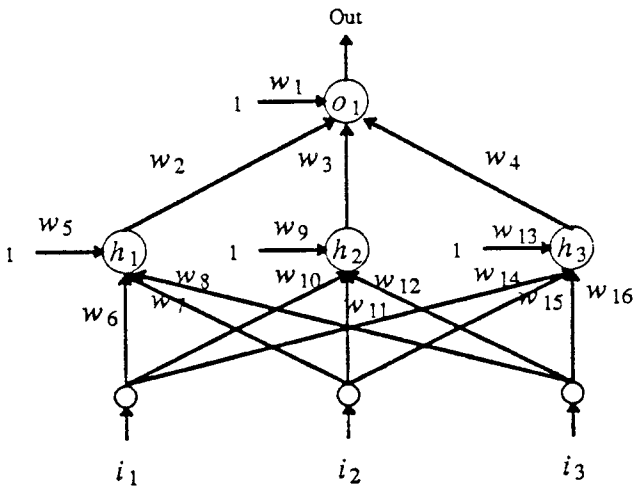


Fig. 8.4 A Network with Three Input Neurons and Three Hidden Neurons

5. SYMMETRIES IN WEIGHT SPACE

An error surface, defined over the weight space of a network, reflects the symmetries of the network and of the desired mapping. These symmetries are very useful to reduce the fraction of the surface that needs to be explicitly explored in order to understand the behavior of the network. In this section we shall investigate the symmetries that exist in our study cases and derive the regions in the weight spaces that need to be investigated.

5.1. Nature of Symmetries

The symmetries exhibited on an error surface of a network can be characterized as those reflecting the symmetries of the network and those reflecting the symmetries of the desired input/output mapping. To see the effects of the symmetries in a network on its associated error surface, let us examine the network in Fig 8.1. The first symmetry in any error surface associated with network 8.1 occurs because the signs of all the weights connected to hidden neuron h_1 can be inverted simultaneously without changing the overall i/o relation of the network [2]. This corresponds to the operation :

opH1 : $w_2 \rightarrow -w_2, w_4 \rightarrow -w_4, w_5 \rightarrow -w_5, w_6 \rightarrow -w_6$.

When we apply opH1 to w, a point in the weight space of network 8.1, opH1w, the image of w under opH1, will have the same error as w, and we say w and opH1w are equivalent according to this type of symmetry. Similarly, changing the signs of the weights connected to neuron $h2$ also leaves the behavior of network 8.1 unchanged; this is achieved by:

opH2 : $w_3 \rightarrow -w_3, w_7 \rightarrow -w_7, w_8 \rightarrow -w_8, w_9 \rightarrow -w_9$.

Since 2^2 different operations can be derived from the combinations of opH1 and opH2, four equivalent configurations result from this type of symmetry. This tells us that the overall input-output relation of a network is unchanged if the signs of all the weights connected to a hidden neuron in that network are simultaneously inverted. Therefore, for a network with H hidden neurons, 2^H equivalent configurations result from this type of symmetry.

For a network with complete connectivity between its layers, the connection patterns of all neurons in the same hidden layer will be identical and any permutation of the roles of the hidden neurons in that layer will lead to the same behavior of the network [2]. For example, for a hidden layer of H neurons where the roles of these H neurons can be arranged in $H!$ different ways, the weight space associated with this case exhibits $H!$ equivalent configurations. In network 8.1, hidden neurons $h1$ and $h2$ are connected identically to the rest of the system, so changing the roles of these two neurons does not affect the function performed by the network. As a result, the error function of network 8.1 is not changed when we make the following four substitutions:

opHx : $w_2 \longleftrightarrow w_3, w_4 \longleftrightarrow w_7, w_5 \longleftrightarrow w_8, w_6 \longleftrightarrow w_9$.

Thus two equivalent configurations result from this symmetry.

When a network with complete connectivity between the input layer and the first hidden layer is used to perform the I-bit parity function, we can arrange I input neurons in $I!$ ways without changing the behavior of the network. These equivalent arrangements result in $I!$ equivalent configurations in the error surface with respect to I-bit parity. For example, when we employ network 8.1 to represent exclusive-OR mapping, the symmetries inherent in this function will be reflected in the error surface. One symmetry in this function comes from the fact that input signals (-1, 1) and (1, -1) both map to (1). Exchanging the identities of input neurons i_1

and i_2 does not affect the global characteristics of the network since the error function sums equally over all input patterns. This exchange can be simulated by the following operation:

opIx : $w_5 \longleftrightarrow w_6, w_8 \longleftrightarrow w_9$.

Since there are 2! ways to arrange two input neurons, two equivalent configurations arise from this type of symmetry.

Another symmetry related to the XOR function occurs because simultaneous complementations of the two inputs do not change the definition of the XOR function. On the other hand, when we complement only one input, the output is complemented. The inversion of the signal to unit i_1 can be simulated by changing the signs of w_5 and w_8. The inversion of the output is achieved by changing the signs of w_1, w_2 and w_3. Therefore, when the operation

opI1 : $w_5 \longleftrightarrow -w_5, w_8 \longleftrightarrow -w_8, w_1 \longleftrightarrow -w_1, w_2 \longleftrightarrow -w_2, w_3 \longleftrightarrow -w_3$

is carried out, the behavior of network 5.2 is left unchanged. In the same way, inversion of the input signal to neuron i_2 and of the output signal according to the operation

opI2 : $w_6 \longleftrightarrow -w_6, w_9 \longleftrightarrow -w_9, w_1 \longleftrightarrow -w_1, w_2 \longleftrightarrow -w_2, w_3 \longleftrightarrow -w_3$

does not affect the behavior of network 8.1. This symmetry results in 2^2 equivalent configurations in the error surface associated with the XOR mapping. When we generalize this result to the case of the I-bit parity function, this type of symmetry produces 2^I equivalent configurations in the associated error surface.

In summary, for a fully connected layered feed-forward network, the number of equivalent configurations produced by a hidden layer of H neurons is $2^H H!$. When such a network is used to represent the I-bit parity function, the error surface associated with this case has at least $2^I I! 2^H H!$ equivalent configurations. For example, when network 8.1 is employed to perform XOR mapping, the associated error surface will exhibit sixty-four[1] fold symmetry. To know these symmetries will enable us to define the fraction of the weight space that we need to explore in order to understand the behavior of such a network.

5.2. Defining a Representative Sector

For any error surface and the weight space over which it is defined, symmetries in the error surface imply certain equivalences among points in the weight space. These equivalences can be used to divide the weight space into regions such that the error surface pieces over these regions have the same characteristics. One such portion of the error surface (which shows all characteristics of the entire error surface) is called a *representative part of the error surface*, and its corresponding domain is called a *representative sector*. We need to explore only a representative sector in order to understand the whole structure of the error surface.

A representative sector can be defined by specifying its boundaries. A logical choice for these boundaries is a subset of the hyper-planes left invariant by inherent symmetry operations. For example, a mirroring operation, $w_i \rightarrow -w_i$, leads to a hyper-plane of the form $w_i = 0$. Exchanging the values of two weights, $w_i \longleftrightarrow w_j$, implies a hyper-plane of the form $w_i - w_j = 0$. Each of these operations reduces the amount of space that must be investigated by a factor of 2; that is, for each operation, we can restrict ourselves to examining only the region to one side of

[1] $2^2 \times 2! \times 2^2 \times 2! = 64$.

the hyper-plane implied by the operation. When more than one operations are involved, hyper-plane boundaries must be properly defined so that the application of all the involved operations to the chosen region can produce the entire weight space.

In selecting hyper-plane boundaries associated with a given set of symmetry operations, we assign the given operations different priorities. Any sector partitioned by hyper-planes associated with operations of higher priorities are closed under operations of lower priorities.[2] For example, if opA, opB are assigned with priorities higher than that of opC, then hyper-planes implied by opA and opB can be defined in such a way that any sector bounded by these two hyper-planes is closed under opC. Once the priorities are determined, the operations of the first priority are used to define hyper-planes to bound a proper region, referred to as the *first-level representative sector*, in an associated weight space. This first-level sector must be defined in such a way that, when we apply all combinations of operations with highest priority to this region, the entire weight space must be generated. We then subdivide the first level sector by the operations of the second priority. The subdivision recursively proceeds in this manner until all the operations have been considered and the final representative sector has been found at the lowest sector level.

Applying the above method to the case of network 8.1 performing the XOR mapping, we assign the first priority to opH1, opH2 and opHx, the operations associated with the symmetries in the network, and the second priority to opI1, opI2 and opIx, the operations associated with the symmetry in the XOR mapping. The reason for this assignment is that opI1, opI2 and opIx do not affect w4 (the threshold of neuron h_1) and w7 (the threshold of neuron h_2). That is, w and its image under any combination of these three operations have the same w4 and w7 components. If we define the hyper-plane associated with opH1, opH2 and opHx in terms of only w4 and w7, any partitions defined by these hyper-planes will be closed under any combination of opI1, opI2 and opIx. We then further subdivide the first level sector by opI1, opI2 and opIx. The second level sector derived from this partition can be used as a representative sector. This second level sector is a valid representative sector, because when we apply all eight distinct combinations of opI1, opI2 and opIx to this region, we produce the first level sector. Then, by applying the eight combinations of opH1, opH2 and opHx to the first level sector, we generate the entire weight space.

To locate the first-level sector, we first employ opH1 and opH2 to divide the weight space into four equivalent sectors. Because the sets of the weights mirrored by each of these two operations are disjoint, we can define hyper-planes opH1 and opH2 independently of each other. The mirroring of w_i implies a hyper-plane of the form $w_i = 0$. OpH1, a composite operation that mirrors w_2, w_4, w_5 and w_6, implies a hyper-plane of the form

$$aw_2 + bw_4 + cw_5 + dw_6 = 0,$$

where a, b, c and d are arbitrary numbers. Similarly, opH2, a composite operation that mirrors w_3, w_7, w_8 and w_9, implies a hyper-plane of the form

$$ew_3 + fw_7 + gw_8 + hw_9 = 0,$$

where e, f, g and h are arbitrary numbers. By restricting ourselves to w_4 and w_7, hyper-planes

---

[2] For function $f : S \rightarrow S$ and subset T in S, f is closed under T if for every x in T, $f(x)$ is also in T.

opH1 and opH2 are reduced to $w_4 = 0$ and $w_7 = 0$. Examining each region bounded by these two hyper-planes, we find that the region that satisfies

$$w_4 \geq 0 \quad \text{and} \quad w_7 \geq 0 \tag{15}$$

is closed under opHx. Since it is closed, we choose this region, implicitly defining the hyper-plane for opHx to be $w_4 = w_7$. Considering these three hyper-planes, we define the first-level sector as a region satisfying

$$w_4 \geq w_7 \geq 0. \tag{16}$$

This is a valid first-level sector, because when we apply all eight combinations of opH1, opH2 and opHx to this region, the entire weight space is generated.

To find second level sectors, we note that the relationship of opI1, opI2 and opIx to one another are similar to that of opH1, opH2 and opH3. Hyper-planes for opI1, opI2 and opIx can thus be chosen as $w_5 = 0$, $w_6 = 0$ and $w_5 = w_6$ respectively. The second level sector which is also a representative sector for a whole problem can be expressed as a region satisfying

$$w_4 \geq w_7 \geq 0 \quad \text{and} \quad w_5 \geq w_6 \geq 0. \tag{17}$$

For a case of a network with H hidden neurons, inequality (16) can be generalized to

$$w_{h1} \geq w_{h2} \geq \cdots \geq w_{hi} \geq \cdots \geq w_{hH} \geq 0, \tag{18}$$

where w_{hi} is the threshold of hidden neuron h_i. Two factors yield these inequalities. The first is that the mirroring of w_{hi} leads to a hyper-plane $w_{hi} = 0$. The second is that one permutation of the threshold values can be selected from the rest by imposing the constraint:

$$w_{h1} \geq w_{h2} \geq \cdots \geq w_{hi} \geq \cdots \geq w_{hH}. \tag{19}$$

Similarly, a sector related to the operations associated with the I-input parity function can be defined as

$$w_{j1} \geq w_{j2} \geq \cdots \geq w_{ji} \geq \cdots \geq w_{jI} \geq 0, \tag{20}$$

where w_{ji} is the weight connecting input neuron i to hidden neuron j. Finally, by combining constraints (19) and (20), a representative sector of a three-layered network with complete connectivity performing the I-input parity function can be defined.

6. THE ERROR SURFACE OF NETWORK 8.1 PERFORMING THE XOR MAPPING

6.1. Region around the Origin

In the error function of network 8.1, with respect to the XOR mapping, the origin is the center of symmetry. Using TsqErr(w) as in Equation 12, the value at the origin is four, and the first derivatives at the origin are zero in all directions. The error surface at the origin has a positive curvature for the axis corresponding to w_1 and zero curvature on all other axes. This is easily understood by looking at the network itself. If all the weights are zero, the output of the network is zero for every input pattern and each individual error is thus one. If we allow only w_1 to be nonzero, the output of the network is

$$O(\mathrm{i}_p) = \frac{1-\exp(-w_1)}{1+exp(-w_1)},$$

for any i_p. Near the origin, this equation can be approximated as

$$O(\mathrm{i}_p) = \frac{1}{2}w_1 - \frac{1}{24}w_1^3.$$

Then, the error near the origin and along w_1 axis can be approximated by:

$$4 + w_1^2.$$

This result tells us that the origin is a local minimum along the direction w_1. Changing any single weight other than w_1 will not result in a change of error, thus producing a surface that is totally flat along all other axes.

Analyzing the pattern of connections in network 8.1, we find that at least four nonzero weights must be involved in defining a direction where the error is reduced when we move away from the origin. A sample direction is

$$(w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9) = (0, 1, 0, -1, 1, -1, 0, 0, 0), \qquad (20)$$

and the error along this direction can be approximated by

$$4.00 - 0.25\varepsilon^4,$$

where ε is the distance from the origin along the direction defined by (20). Thus the origin of the error surface of network 8.1 in representing the XOR function is a multidimensional saddle point. Close inspection shows that it is a *plateau* in which a few very shallow valleys start radially from the origin; in all other directions the surface turns upwards.

6.2. Minima in the Error Surface

To obtain a survey of the minima in the error surface of network 8.1 in performing the XOR mapping, we employeded a search from one thousand points randomly distributed in the representative sector. We limited the value of each weight to [-3, 3], reducing the associated weight space to a compact set, and recorded the local minima found in this 9-dimensional cube. All minima fall on the surface of this cube, i.e. the boundary of the weight domain. For each minimum, we also recorded the percent of the steepest descents approaching that point. This statistical figure reflects the fraction of the weight space covered by the collection zone of that

minimum.

The minima found can be separated into three types; the minima that lie in the representative sector defined by Inequalities (17) are used as representatives of their types and are listed in Table 1. Because of the 64-fold symmetry inherent in this error surface, each such minimum will be replicated sixty-four times; however, some of these replicas may fall on top of one another. For example, minimum MA has a multiplicity of four and thus there are a total of only sixteen such locations in the whole weight space.

The minima of type MA and MB are local minima of distinct points while the minima of type MC are not. Each member in type MC is a connected set that forms a *groove* of constant error; the same minimal error is obtained for all points where the sum of two weights w_2 and w_3 is equal to 1.62. These points satisfy the conditions

$$w_4 = w_7, \quad w_5 = w_8, \text{ and } w_6 = w_9, \tag{21}$$

and

$$w_2 + w_3 = 1.62. \tag{22}$$

Condition (21) implies that neurons h_1 and h_2 have identical roles and thus their outputs are the same for every input. Denoting these two equal outputs by s, the output of network 8.1 becomes $swf\,((w_2 + w_3)s)$, where $swf\,()$ is the sigmoid function. According to condition (22), for every point in the groove, the output of network becomes $swf\,(1.62s)$.

Examining the minima in Table 1, we see that all of them lie at the surface of the 9-dimensional cube of half-size 3. They are not the real minima on the whole error surface; they are simply the lowest points in valleys that radiate out from the center of the surface and get truncated by the finite size of the domain. To find the real minima, we can selectively trace these valleys to larger distances from the origin.

Based on the capability of a minimum in representing a solution to the XOR task, we classify these minima into two classes: *good* and *bad* minima. Good minima produce the desired output value of the XOR function for all four possible input patterns within some small error margin; while bad minima produce an output that will show an unacceptably large error for one or more input patterns. With this classification, MA can be considered a good minimum, while MB and MC are bad minima.

Table 1
Minima in Error Surface of Network 8.1
Performing the XOR Mapping

Identifier	MA	MB	MC
Valley	AC	B	AC
Error	0.0803	2.0969	2.8835
Radius from the Origin	8.95	7.42	7.48*
Multiplicity	4	2	4
Percent of the Weight Space Covered by the Collection Zone of the Minimum	60.3	37.5	2.2
w_1	2.86	0.00	0.82
w_2	-3.00	3.00	A
w_3	-3.00	-3.00	B
w_4	3.00	3.00	3.00
w_5	3.00	3.00	3.00
w_6	-3.00	1.05	-3.00
w_7	3.00	3.00	3.00
w_8	-3.00	3.00	3.00
w_9	3.00	-1.05	-3.00
Classification	good	bad	
Percent of the Weight Space Covered by the Collection Zone of the class	60.3	39.7	

A + B = -1.62

* minimum radius

6.3. Characteristics of Valleys

Since the valleys run roughly radially away from the origin, we characterize them by studying their error value profiles on subsequent spheres of larger and larger radii around the origin. On each 9-dimensional sphere, we determine a local minimum by running a steepest descent from a starting point that is given by the projection of the local minimum on the previous sphere. Finding the local minimum has been implemented as follows:

Assume that at iteration n, the process is at point $\mathbf{x}_n$. To find $\mathbf{x}_{n+1}$, we first compute the gradient of the error function at $\mathbf{x}_n$. This gradient is projected along the radius direction onto the 8-dimensional hyper-plane which is tangent to the sphere at $\mathbf{x}_n$. Let us denote the projected gradient by $(\nabla \mathrm{Err})_p$. According to this projected gradient, the process moves downhill to a point ($\mathbf{x} + h(\nabla \mathrm{Err})_p$), where h is a step-size. Projecting the point ($\mathbf{x} + h(\nabla \mathrm{Err})_p$) along its radius direction back onto the sphere locates point $\mathbf{x}_{n+1}$. This iterative process stops when $(\nabla \mathrm{Err})_p$ is smaller than a given threshold. At our stopping point, we define an 8-dimensional hyper-plane tangent to the sphere and then compute the principal curvatures of the error function on this hyper-plane. If all eight curvatures are positive, the stopping point is a local minimum on the sphere and the process ends. In all other cases, we choose a new point in the vicinity of the stopping point and lying on the axis of strongest negative curvature. From this point, we again run the steepest descent to locate a new stopping point. We continue this process until we find a final local minimum.

We now project this minimum point along its radius direction onto a new sphere of larger radius and locate a minimum on the new sphere with the above process. Generally, this will quickly converge to a corresponding minimum on the new sphere, since the minima on the two spheres are near each other. The locus of all the minima on subsequent spheres defines the *floor of a valley*.

To explore the structure of the valley associated with minimum MA, we run the above process from minimum MA tracing out the valley's floor backwards toward the origin. This floor is then used to represent the *path* of the valley. In Figure 9 we portray the path of this valley by plotting the coordinates w_1 ,..., w_9 of the local minima as a function of the radius of the corresponding sphere. Near the origin, the valley splits into two branches which then merge again at a distance of about 2 from the origin. We display the *profile* of the valley by plotting the error value along the floor of the valley as a function of the radius (graph AC in Figure 10). This graph illustrates that the floor of the valley approaches 0 if we follow it far enough from the origin. It is thus a *good* valley since it leads to a proper solution for the XOR problem.

We also apply this method to examine the valley associated with minimum MB. We find that when the radius of the sphere is less than 3.3, there is only one type of minimum located on a sphere, the one that belongs to the valley associated with minimum MA. We therefore say that the valley associated with minimum MB starts only at a radius of about 3.3. Graph *B* in Figure 10 portrays the profile of this valley. Since the floor of this valley never falls below an error value of 2.0, it represents a *bad* valley that never yields a solution.

For the valley associated with minimum MC, we find that the local minimum located by a steepest descent on each sphere is the same as that of the valley associated with minimum MA. We therefore conclude that minima MA and MC lie in the same class of valley. The reason that

two separate minima, MA and MC, can be associated with one valley can be understood by looking at Figure 11. This figure shows the contour plot of a scalar-valued function defined over two-dimensional space. When the domain of this function is restricted to the region inside the square box, points M1, M2 and M3 become local minima of the restricted function. If, however, we run a steepest descent from the minima M1, M2 and M3 while limiting the search space to the circle shown, then these three steepest descents all stop at the same point on the circle labeled Min. This figure is an oversimplification, but it provides a model for the real situation of minima MA and MC.

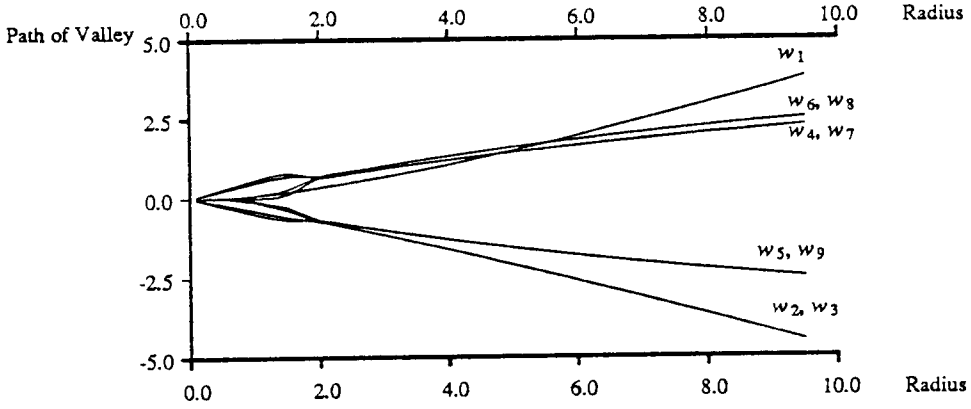


Fig. 9 The Locus of the Floor of the Valley Associated with Minimum MA in Table 1.

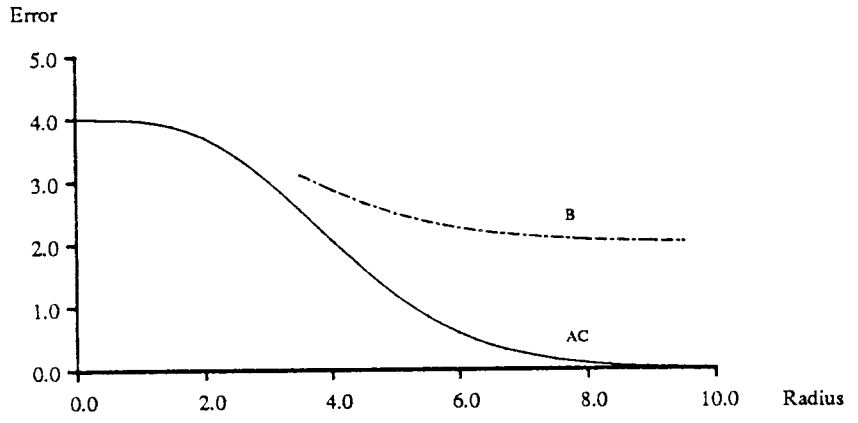


Fig. 10 The Profiles of Valleys on the Error Surface of Network 8.1 Performing the XOR Mapping.

To obtain some information about the *shape* of the valleys, we studied their cross-section on the 9-dimensional spheres around the origin. We first located the floor of a valley on a

particular sphere. Next, we defined an 8-dimensional hyper-plane tangent to the sphere at the intersection point. By restricting the domain of the error function to this hyper-plane, we derived eight principal curvature axes from the restricted function. Figure 12 shows the error values along an arc produced by the intersection of the plane through the principal curvature axis and the origin; the lateral deviation is measured in degree from the floor of the valley along the arc.

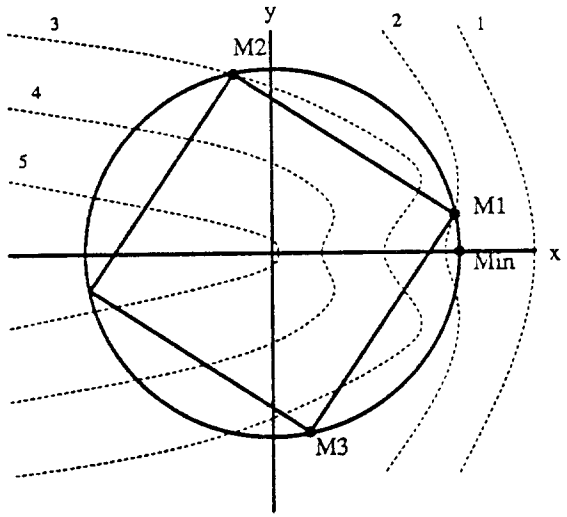


Fig. 11 A Conceptual Model Illustrating that Limitation of All Weights to Fixed Intervals Can Lead to Multiple Minima.

Next, we have tried to obtain some measure for the cross-section of a valley as a function of R, the distance from the origin. For each error value profile on a particular sphere (Fig. 12), we locate the interval (θ_{upper}, θ_{lower}) where the error is less than four, i.e., the error value at the origin. We use the smaller value of these two bounds for each axis and call it the *half –width*. Using all the eight half-widths, we calculate the area of an 8-dimensional elliptical patch providing some measure of a cross-section of a valley.

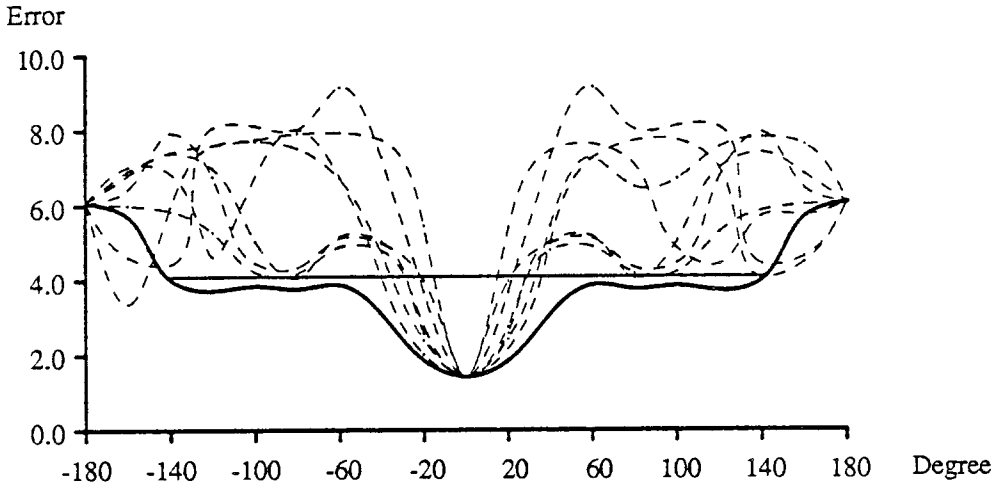
$$\text{Area of elliptical patch} \quad = \quad \frac{\pi^4}{24} R^8 \left(\prod_{i=1}^{8} \theta_i\right),$$

where R is a radius of the sphere and θ_i (measured in radians) is the half-width associated with axis i. The cross-sectional areas of these valleys as functions of a sphere's radius are displayed in Figure 13. Our measure for the cross-section of valley AC displays a discontinuity at a radius of about 4.8. This can be understood by comparing the structure of the cross-section at radius 4.7 (Fig. 12a) with that at radius 4.9 (Fig. 12b). Around a radius of 4.8, the shoulders of one of the profiles (shown as a solid line) rise above 4.0 and thus lead to a drastic reduction in the cross-section. Figure 14 shows the graph of the product of all the eight angular half-widths ($\prod_{i=1}^{8} \theta_i$) as

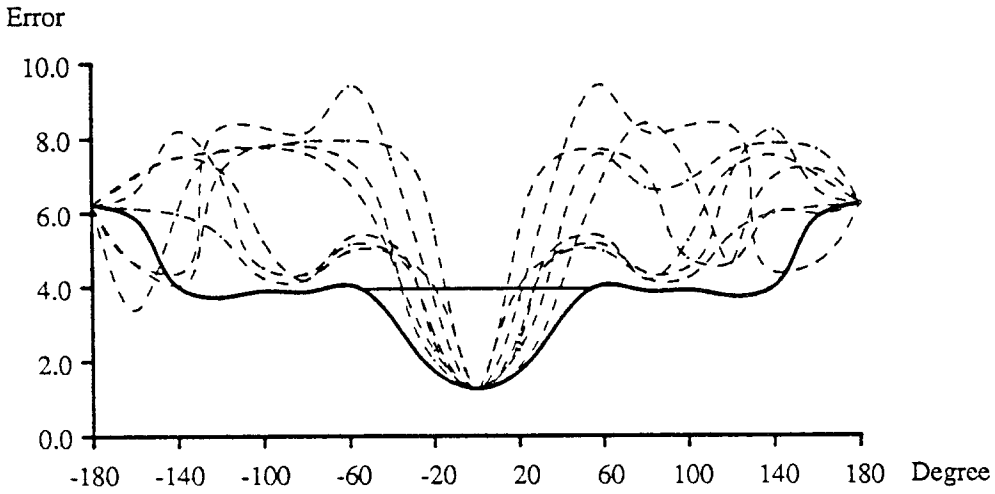
---

The volume of an n-dimensional sphere with radius R is equal to $\frac{2(\pi^{n/2})R^n}{n\,\Gamma(\frac{n}{2})}$ where $\Gamma(\frac{n}{2})$ is defined to be $\frac{\sqrt{\pi}(n-2)(n-4)\cdots 1}{2^{(n-1)/2}}$ when n is odd, and $(\frac{n-2}{2})!$ when n is even.

a function of a sphere's radius, thus providing a measure of the fraction of the area of the total sphere that has an error value less than 4.0. This view which is more sensitive in the region around the origin shows a second discontinuity in valley AC at a radius of about 2.0. This occurs because two valleys radiating from the origin merge into a single valley at this point (see Fig. 9).



(a)



(b)

Fig. 12 Cross-Section of the Valley Associated with Minimum MA :
(a) on the Sphere with Radius 4.7,
(b) on the Sphere with Radius 4.9.

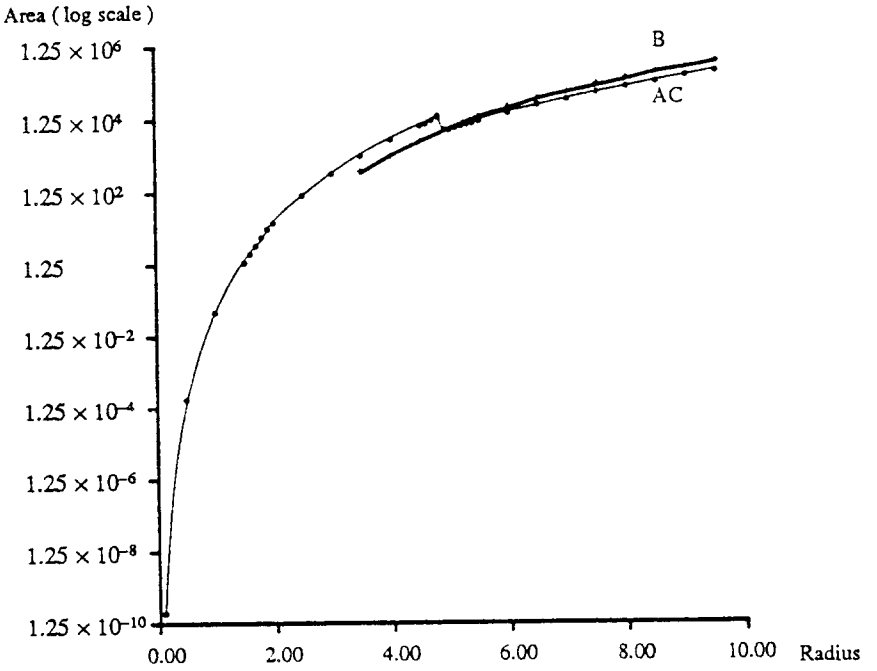


Fig. 13 The Cross-sectional Areas of the Valleys
on the Error Surface of Network 8.1 Performing the XOR Mapping.

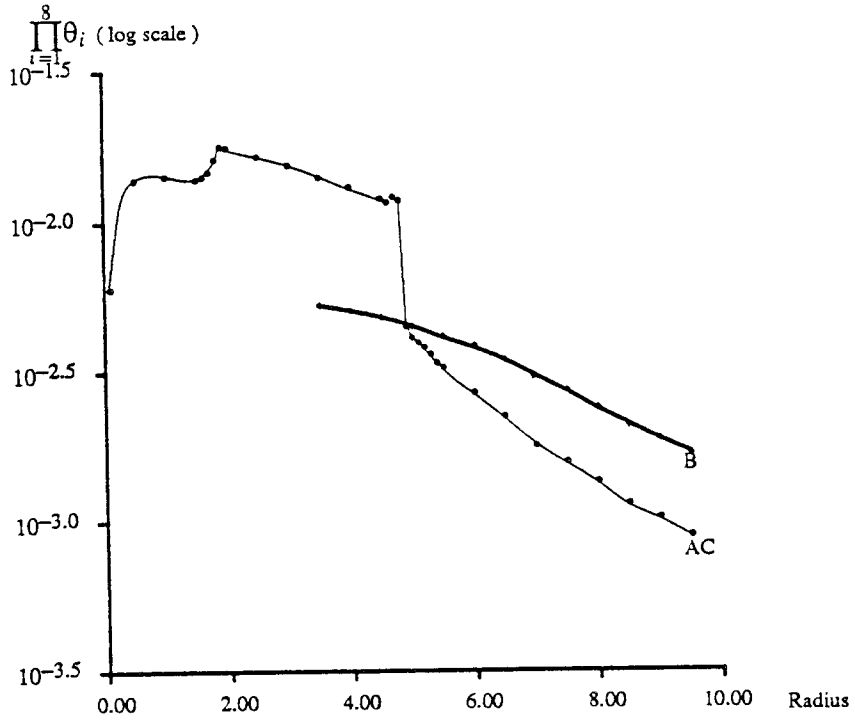


Fig. 14 The Product of all the Eight Half-widths ($\prod_{i=1}^{8}\theta_i$) of the Valleys on the Error Surface of Network 8.1 Performing the XOR Mapping.

7. ERROR SURFACES OF OTHER NETWORKS PERFORMING THE XOR MAPPING

7.1. The Error Surfaces of Networks 8.2 and 8.3 Performing the XOR Mapping

To gain insight into changes in the structures of an error surface as the number of hidden neurons in a network increases, we study two other networks implementing the XOR function; one has three hidden neurons (Fig. 8.2), and the other has four (Fig. 8.3). Network 8.2 differs from network 8.1 by the additional hidden neuron h_3 and its associated weights w_4, w_{11}, w_{12} and w_{13}. Similarly, network 8.3 has an additional hidden neuron h_4 and weights w_5, w_{15}, w_{16} and w_{17}. To investigate the error surfaces of networks 8.2 and 8.3, we have employed the approaches discussed above for network 8.1.

The characteristics of the region around the origins of the error surfaces of networks 8.2 and 8.3 are similar to those of network 8.1. The errors at the origins of the error surfaces of network 8.2 and 8.3 are 4.0, and the first derivatives at their origins are zero in every direction. We also find that both surfaces have positive curvatures along the w_1-axis, and that near the origin the error along this axis can be approximated by

$$4 + w_1^2.$$

To confirm that the two functions again form saddle points, we look back to the case of network 8.1. If the weight space of network 8.2 is limited to $\mathbf{W}_{8.1}$, i.e., the subspace spanned by w_1, w_2, w_3, w_5, w_6, w_7, w_8, w_9, and w_{10}, the error surface of network 8.2 will be identical (except for the labeling of the coordinates) to the error surface of network 8.1. This likeness tells us that the error surface of network 8.1 is a subset of the error surface of network 8.2 when both networks represent the same function. In section 6.1, we have shown that the error surface of network 8.1 has a direction where the error is reduced when we move away from the origin. Because the error surface of network 8.1 is a subset of the error surface of network 8.2, one will find the same behavior along this direction in the error surface of network 8.2. We can therefore conclude that the origin of the error surface of network 8.2 is also a multi-dimensional saddle point. By the same process, one can also conclude that the origin of the error surface of network 8.3 is a multi-dimensional saddle point.

By probing the weight space with random starting points, followed by gradient descent, we find four types of minima, denoted by MD, ME, MF and MG, in the error surface of network 8.2, and eight types of minima, denoted by MH, MI, MJ, MK, ML, MM, MN and MO, in the error surface of network 8.3. The representatives of these minima for networks 8.2 and 8.3, which are points in the representative sectors, are listed in Tables 2 and 3, respectively.

In network 8.2, the minima of type MD are distinct points while those of types ME, MF and MG are sets of points with the constant error values. ME and MG are again one dimensional grooves, while MF is a two dimensional surface. One interesting point about minima MF is that every point in this type of minima satisfies

$$w_1 + \frac{w_4(1 - exp(w_{11}))}{1 + exp(w_{11})} = -2.86.$$

The expression on the left hand side of this equation is simply the threshold of the output neuron which is modified by adding to weight w_1 a constant of

$$\frac{w_4(1-exp(w_{11}))}{1+exp(w_{11})}.$$

This constant comes from the fact that w_{12} and w_{13} of every point in minima MF are zero. As a result, the signal contributed by hidden neuron h_3 to the output neuron is a constant, thus acting like an additional threshold of the output neuron.

Comparing the data in Tables 1 and 2, we find that the minima MF contain the minimum point MA If we choose the point in minima MF where the weights w_4, w_{11}, w_{12}, and w_{13} that are absent in network 8.1 are all zero, w_1 becomes -2.86, and all the corresponding weights of the two networks are the same. We also find that minima MG in the case of network 8.2 is a transformed version of minima MB. If we allow w_2 of network 8.1 to be larger than 3.0 while still restricting the other weights in the range of [-3, 3], minimum MB will move to the point

$$(w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9)$$
$$= (-0.10,\ 3.10, -3.00,\ 3.00,\ 3.00,\ 1.04,\ 3.00,\ 3.00, -1.03).$$

This new minimum is equivalent to the point in minima MG where $w_2 = 3.10$ and $w_4 = 0.0$. Hence, minimum MB is transformed into a minimum of class MG when we add an extra hidden neuron to network 8.1, and it remains a projection of MG onto the subdomain of network 8.1.

In network 8.3, minima of types MH, MJ, MK and MN are distinct points, while those of types MI, ML can be viewed as grooves of constant error, and minima MM and MO are planes of constant error. Comparing the data in Table 2 and 3, we find that minima ML, MM, MO contain minima MD, ME, and MG, respectively.

The characteristics of the valleys on the error surface of network 8.2 are illustrated in Figures. 15, 16, and 17. There are only two types of valleys on this error surface and both of them are good valleys. MD, ME, and MG are all associated with the same valley (valley DEG in Fig. 15). The cross-sectional areas of these two valleys are displayed in Figure 16, and the product of all the thirteen angular half-widths ($\prod_{i=1}^{13}\theta_i$) is shown in Figure 17. The cross-section of the main valley shows several discontinuities. The ones at radius 2.0 and 4.8 are the ones already known from network 8.1 (Fig. 14). New discontinuities at radii of about 7 and about 13 are caused by the merging of two valleys into one and by an abrupt change of a half-width, respectively.

The characteristics of the valleys on the error surface of network 8.3 are displayed in Figures 18, 19, and 20. One point about valley M1 deserves mentioning: no minimum lies inside valley M1, because valley M1 merges into valley HLO at a radius of about 7.5. Thus, a steepest descent that runs into valley M1 will eventually slide into valley HLO, and stop at a minimum in valley HLO.

Table 2
Minima in the Error Surface of Network 8.2
Performing the XOR Mapping

Identifier	MD	ME	MF	MG
Valley	DEG	DEG	F	DEG
Radius from the Origin	10.07	10.22 *	8.93 *	8.42 *
Error	0.0605	0.0665	0.0803	2.0965
Multiplicity	2	2	8	2
Percent of the Weight Space Covered by the Collection Zone of the Minimum	67.1	25.3	1.5	6.1
w_1	-2.20	-3.00	C	-0.10
w_2	3.00	A	3.00	F
w_3	3.00	B	3.00	-3.00
w_4 +	-1.01	3.00	D	G
w_5	3.00	3.00	3.00	3.00
w_6	3.00	3.00	3.00	3.00
w_7	2.70	3.00	3.00	1.04
w_8	3.00	3.00	3.00	3.00
w_9	-2.70	3.00	-3.00	3.00
w_{10}	-3.00	3.00	-3.00	-1.03
w_{11} +	3.00	2.92	E	3.00
w_{12} +	3.00	-3.00	0.00	3.00
w_{13} +	-3.00	-3.00	0.00	1.04
Classification	good			bad
Percent of the Weight Space Covered by the Collection Zone of the class	93.9			6.1

$A + B = 3.42$
$C + D\left(\frac{1-\exp(-E)}{1+\exp(-E)}\right) = -2.86$
$F + G = 3.10$
* minimum radius
+ new over network 8.1

Table 3
Minima in the Error Surface of Network 8.3 Performing the XOR Mapping

Identifier	MH	MI	MJ	MK	ML	MM	MN	MO
Valley	HLO	IK	J	IK	HLO	M	N	HLO
Error	0.0005	0.0213	0.0236	0.0356	0.0605	0.0665	2.0808	2.0965
Radius from the Origin	12.00	11.09*	11.22	11.40	10.96*	11.38*	10.45	9.41*
Multiplicity	8	4	16	2	4	12	8	6
Percent of the Weight Space Covered by the Collection Zone of the Minimum	48.6	4.5	5.7	38.6	1.4	0.5	0.2	0.5
w_1	0.00	-3.00	-3.00	-2.97	-2.20	-3.00	0.00	-0.10
w_2	3.00	A	3.00	3.00	3.00	E	3.00	H
w_3	3.00	B	3.00	3.00	3.00	F	3.00	I
w_4	-3.00	3.00	3.00	3.00	C	G+	-3.00	J+
w_5	-3.00	-3.00	3.00	-3.00	D+	3.00	-3.00	-3.00
w_6	3.00	3.00	1.49	3.00	3.00	3.00	3.00	3.00
w_7	3.00	3.00	3.00	3.00	3.00	3.00	3.00	3.00
w_8	3.00	3.00	3.00	1.18	2.70	3.00	0.54	1.04
w_9	3.00	3.00	1.49	3.00	3.00	3.00	3.00	3.00
w_{10}	-3.00	3.00	3.00	-3.00	-2.70	3.00	3.00	3.00
w_{11}	-3.00	3.00	3.00	-3.00	-3.00	3.00	0.54	1.04
w_{12}	3.00	2.82	1.49	3.00	3.00	3.00+	3.00	3.00+
w_{13}	3.00	-3.00	-3.00	3.00	3.00	3.00+	3.00	3.00+
w_{14}	-3.00	-3.00	-3.00	1.18	-3.00	3.00+	-0.54	1.04+
w_{15}	3.00	0.68	1.49	3.00	3.00+	2.92	3.00	3.00
w_{16}	-3.00	0.84	-3.00	3.00	3.00+	-3.00	3.00	3.00
w_{17}	3.00	0.84	-3.00	-1.17	-3.00+	-3.00	-0.54	-1.03
Classification	good						bad	
Percent of the Weight Space Covered by the Collection Zone of the class	99.3						0.7	

A + B = 5.32; C + D = -1.01; E + F + G = 3.42; H + I + J = 3.10
* minimum radius
+ new over network 8.2. Note that the added weights may be different from w_5, w_{15}, w_{16} and w_{17}.
This occurs because the points listed in this table are the points in the representative sector.
Weight w_5, w_{15}, w_{16} and w_{17} are mapped to other weights when it is mapped into the representative sector.

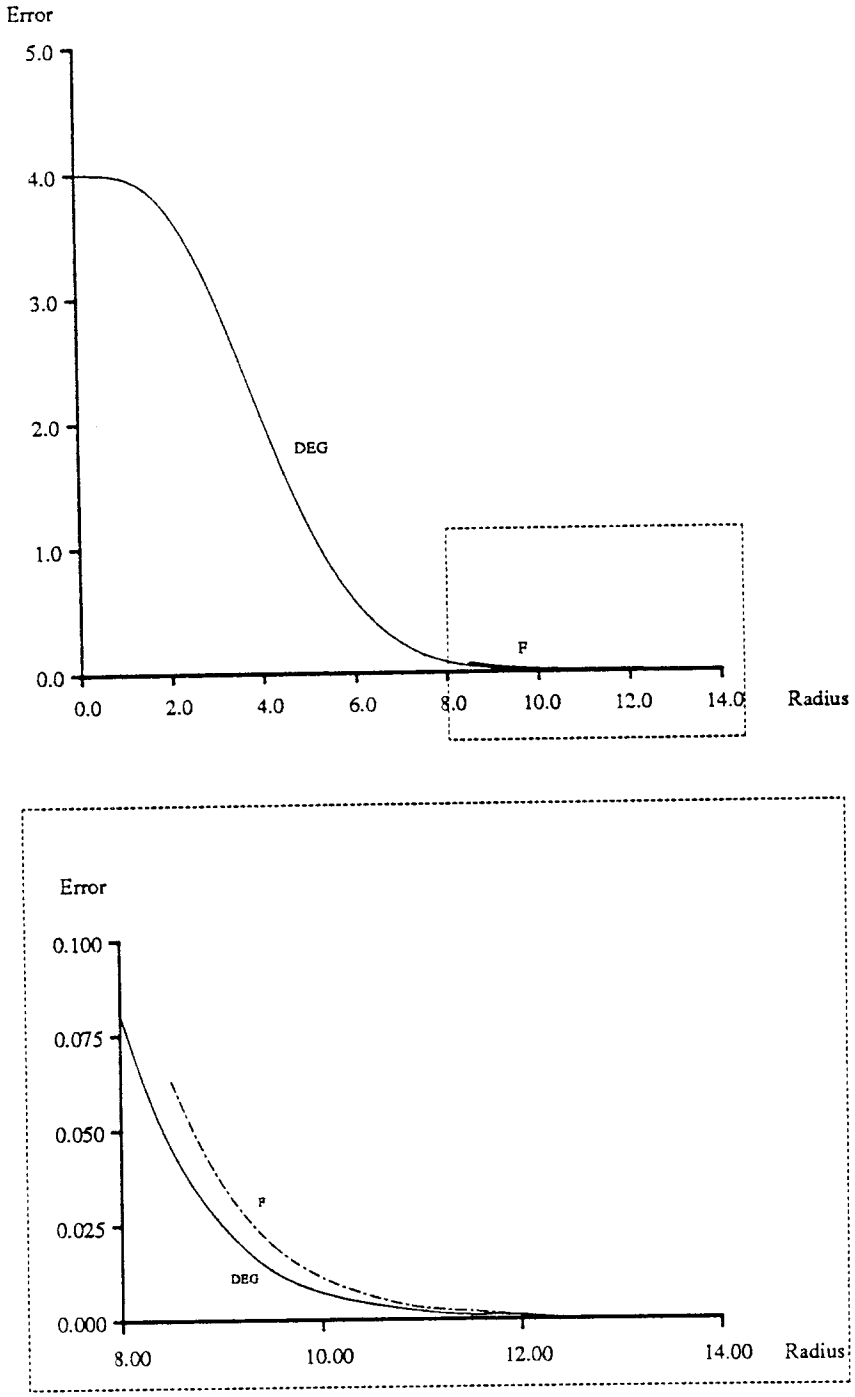


Fig. 15 The Profiles of Valleys on the Error Surface of
Network 8.2 Performing the XOR mapping.
At the Bottom Is a Magnified Image of the Indicated Portion of the Graph Shown on the Top.

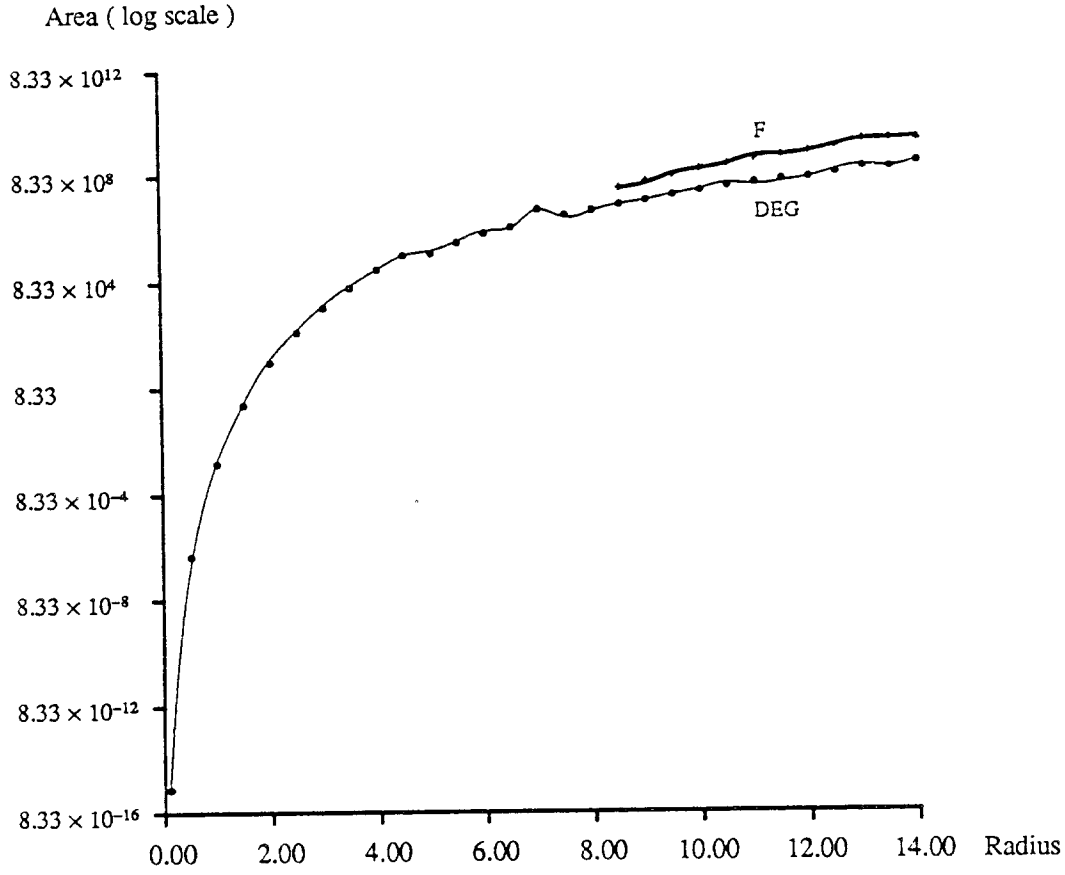


Fig. 16 The Cross-sectional Areas of Valleys
on the Error Surface of Network 8.2 Performing the XOR Mapping.

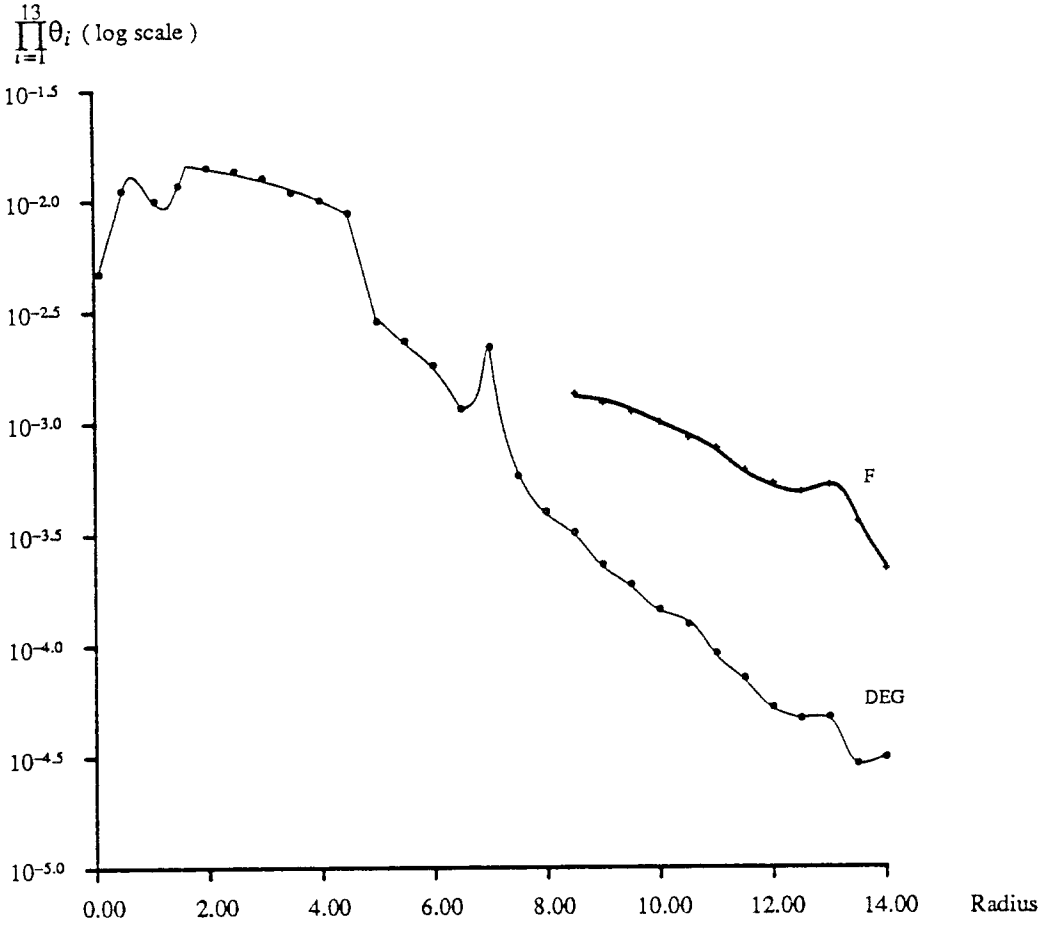


Fig. 17 The Product of All the Thirteen Half-widths ($\prod_{i=1}^{13}\theta_i$) of Valleys on the Error Surface of Network 8.2 Performing the XOR Mapping.

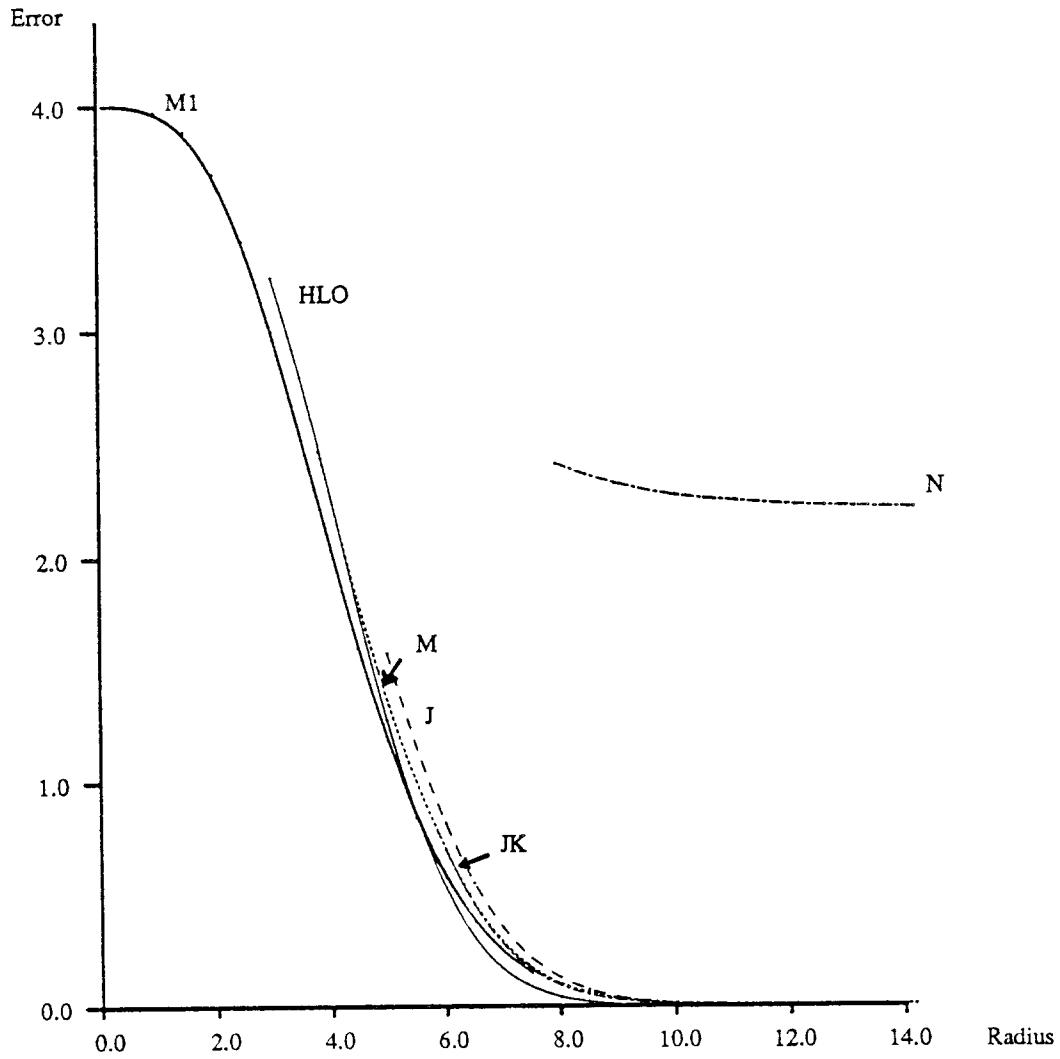


Fig. 18 The Profiles of Valleys on the Error Surface of Network 8.3 Performing the XOR Mapping.

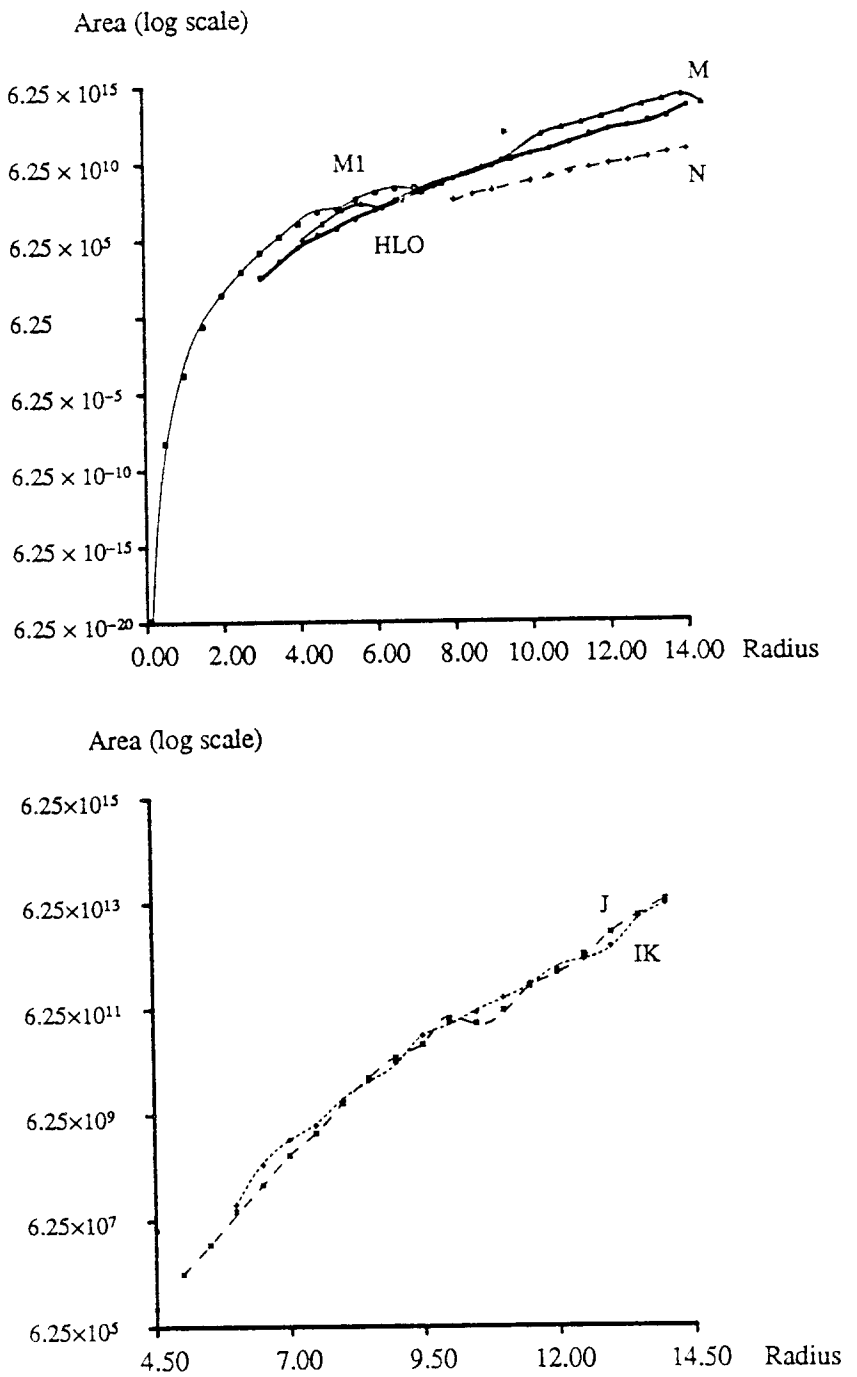


Fig. 19 The Cross-sectional Areas of the Valleys
on the Error Surface of Network 8.3 Performing the XOR Mapping.

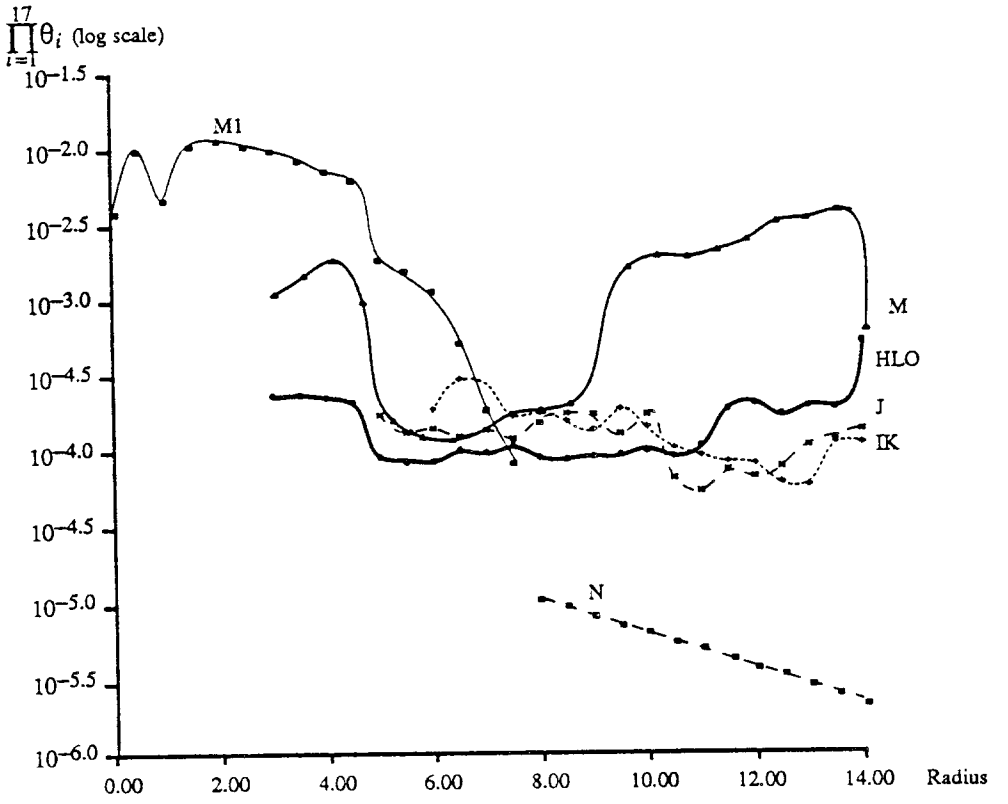


Fig. 20 The Product of All the Seventeen Half-widths ($\prod_{i=1}^{17}\theta_i$) of Valleys on the Error Surface of Network 8.3 Performing the XOR Mapping.

8. ERROR SURFACE OF NETWORK 8.4 PERFORMING 3-INPUT PARITY MAPPING

Investigating the error surface of network 8.4, computing the three-input parity function, we find that the error at the origin is 8 and the first derivatives at the origin are zero in every direction. The curvatures of the surface at the origin are zero except for the curvature along axis w_1. The error near the origin along axis w_1 can be approximated by

$$8 + 2w_1^2.$$

At the origin we find that along the direction

$$(w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9, w_{10}, w_{11}, w_{12}, w_{13}, w_{14}, w_{15}, w_{16})$$
$$= (0, 1, 0, 0, -1, 1, 1, -1, 0, 0, 0, 0, 0, 0, 0, 0), \qquad (23)$$

the error can be approximated by

$$8 - 10\varepsilon^4,$$

where ε is a distance from the origin along the direction defined by (23). Because there is a down-hill path at the origin, the error surface has a saddle point at the origin.

Employing the search method used in the case of network 8.1, we found nine types of minima. The representatives of these nine types are listed in Table 4. This table tells us that minima of types MP, MQ, MR, MV and MW are distinct points, while those of types MS, MT and MU are grooves of constant error values. On this error surface, we found three valleys, the characteristics of which are illustrated in Figures 21, 22 and 23. Figure 21 indicates that of the three valleys, two are good, since their floors approach zero when one proceeds far enough from the origin. The good valley (P-X), associated with minima MP, MQ, MS, MT, MU, MV and MX, starts at the origin. The second good valley (R), associated with minimum MR, starts at a distance of about 7.0 from the origin. The valley (W), associated with minimum MW, starts at a distance of about 6 from the origin. Its floor is not close to zero; it is therefore a bad valley.

Table 4
Minima in Error Surface of Network 8.4
Performing 3-input Parity Mapping

Identifier	MP	MQ	MR	MS	MT	MU	MV	MW	MX
Valley	P-X	P-X	R	P-X	P-X	P-X	P-X	W	P-X
Error	0.1551	0.1661	0.7123	3.3473	3.4420	4.1937	5.9983	6.0635	7.1144
Radius from the Origin	9.63	9.49	9.20	9.22*	8.24*	8.42*	8.92	8.27	7.53
Multiplicity	4	8	12	12	24	8	4	6	2
Percent of the Weight Space Covered by the Collection Zone of the Minimum	85.2	1.5	8.7	0.3	1.3	2.3	0.2	0.3	0.2
w_1	0.00	0.00	0.00	-1.60	0.00	0.00	-2.74	-3.00	1.30
w_2	-3.00	3.00	3.00	3.00	C	E	3.00	1.84	2.14
w_3	-3.00	3.00	-3.00	A	D	F	3.00	1.84	-0.95
w_4	-3.00	-3.00	-3.00	B	-3.00	-3.00	-3.00	1.84	-3.00
w_5	0.00	0.00	3.00	3.00	0.00	0.00	3.00	3.00	3.00
w_6	3.00	3.00	1.45	1.75	3.00	3.00	1.77	1.69	1.56
w_7	3.00	0.00	1.45	1.75	3.00	3.00	1.13	1.51	1.56
w_8	2.99	0.00	1.45	1.75	3.00	1.05	1.13	1.51	0.32
w_9	0.00	0.00	3.00	0.82	0.00	0.00	3.00	3.00	3.00
w_{10}	1.18	-3.00	-1.45	3.00	3.00	3.00	1.77	-1.51	-1.25
w_{11}	-1.18	-3.00	-1.45	3.00	3.00	3.00	-1.13	1.51	-1.25
w_{12}	-3.00	-3.00	-1.45	3.00	3.00	1.05	-1.13	-1.69	-1.09
w_{13}	0.00	0.00	0.00	0.82	0.00	0.00	2.67	3.00	3.00
w_{14}	-3.00	3.00	3.00	3.00	-0.78	-3.00	2.94	-1.51	1.43
w_{15}	-3.00	-3.00	3.00	3.00	-0.78	-3.00	0.00	-1.69	1.43
w_{16}	2.99	-3.00	3.00	3.00	-0.78	1.05	0.00	1.51	-0.44
Classification	good			bad					
Percent of the Weight Space Covered by the Collection Zone of the class	95.4			4.6					

A + B = -2.53; C + D = 2.47; E + F = -3.01
* minimum radius

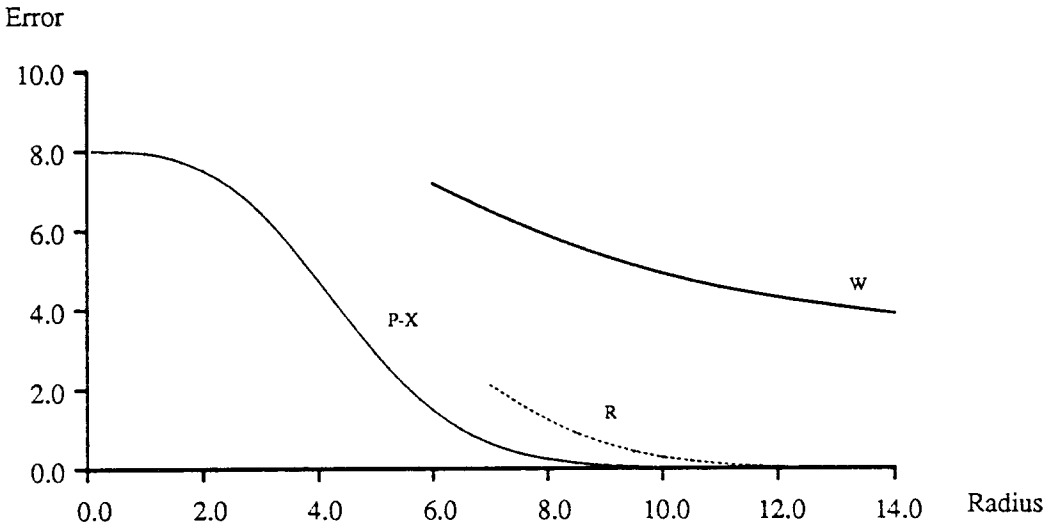


Fig. 21 The Profiles of Valleys on the Error Surface of Network 8.4 Performing the 3-Input Parity mapping.

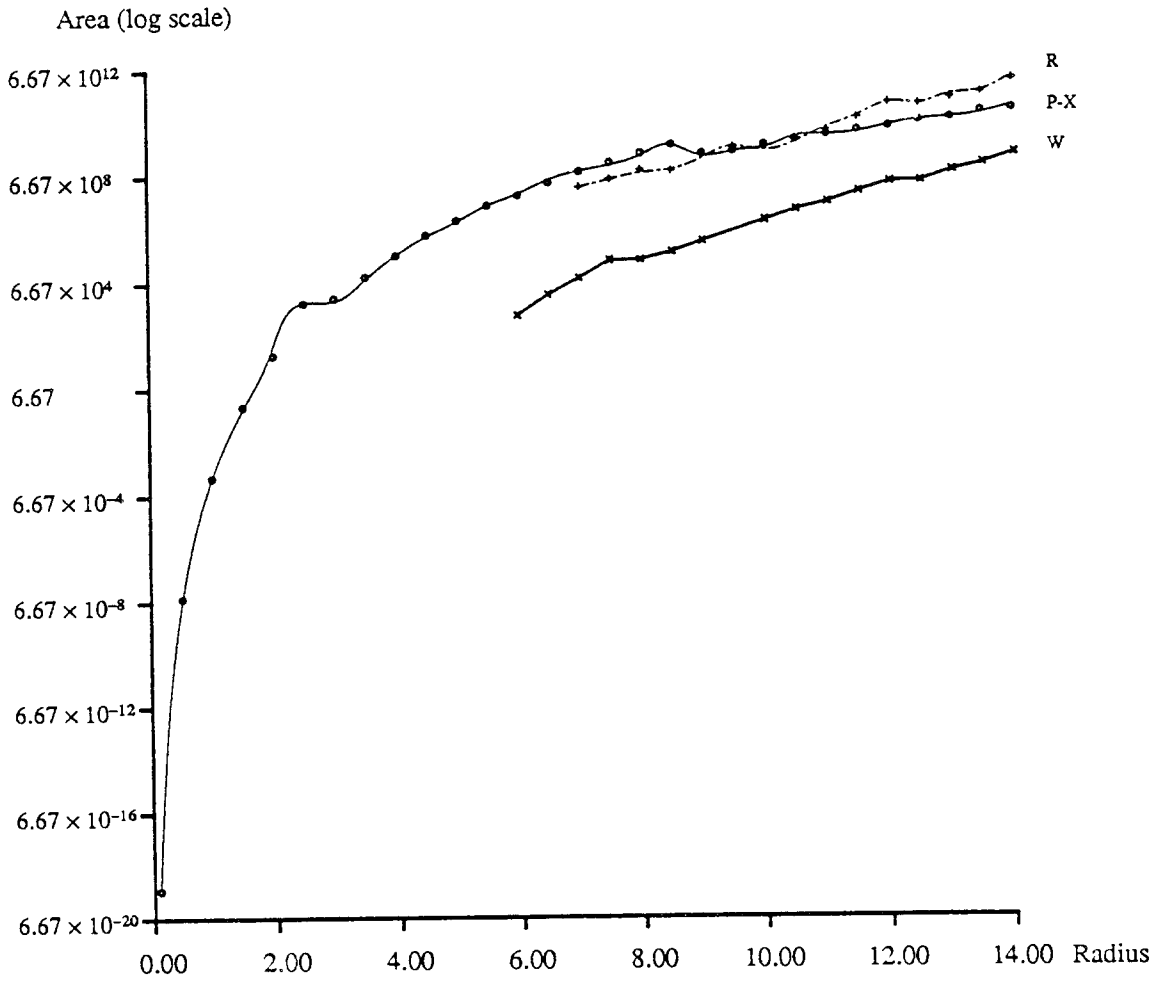


Fig. 22 The Cross-sectional Areas of Valleys
on the Error Surface of Network 8.4 Performing the 3-Input Parity Mapping.

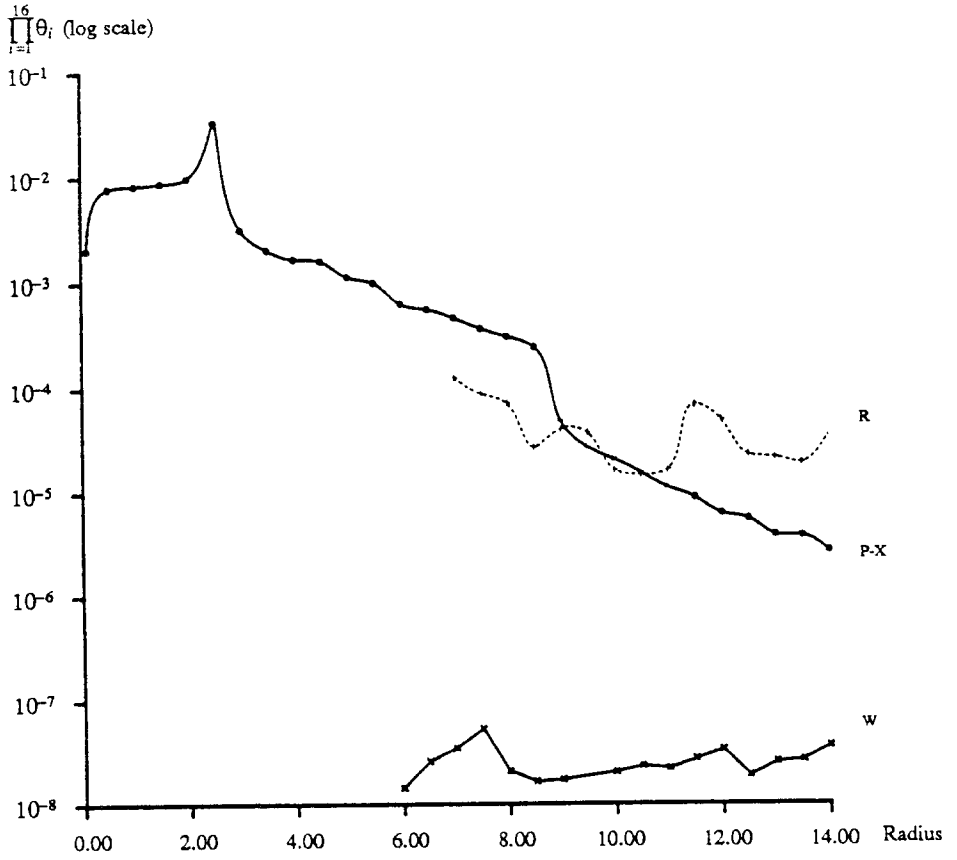


Fig. 23 The Product of All the Sixteen Half-widths ($\prod_{i=1}^{16}\theta_i$) of Valleys on the Error Surface of Network 8.4 Performing the 3-Input Parity Mapping.

9. WEIGHT CONTAINMENT

Our study of some sample error surfaces have shown that all minima on these surfaces are at infinity. This characteristics is not desirable because actual weights are typically limited to finite values. There are several methods to produce stable, good minima at the finite distance from the origin. Besides limiting the weight space to be a compact set, we can move the locations of all minima to a finite distance from the origin by using weight decay or by setting the goal values of the output function to be 0.9 or -0.9 rather than ±1.0. The update formula with weight decay is:

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \eta \nabla Err(\mathbf{w}) - \delta \mathbf{w},$$

where η and δ are some constants. [8] recommends δ for the XOR task to be 0.0001.

To see the correlation among the three weight containment methods mentioned above, we run a steepest descent from each minimum on the error surface of network 8.1 performing the XOR task. These three weight containments are used to prevent the steepest descent from going too far from the origin. Since the minima of class MC are not distinct points, we chose three separate points: the point where $w_2 = w_3 = -0.81$, the point where $w_2 = -1.62$ and $w_3 = 0.0$, and the point where $w_2 = 0.0$ and $w_3 = -1.62$. Geometrically, the first point is a symmetry point in the middle of the groove, while the other two points are more general, asymmetric solutions. The data gained from this experiment are illustrated in Table 5. This data indicates that the results derived from all three weight containments are consistent; if one scheme locates a good minimum, the other schemes will also locate a corresponding good minimum.

In order to compare these results with the minima found by limiting the weight domain, we also terminated the steepest descent from each minimum at the surface of a cube of half-size 5 (C-5) and also with a sphere of radius 18 (R-18). Examining the minima at R-18, we find that the ones corresponding to points MA and MB have error values of 0.000001 and of 2.000014, respectively. This data is consistent with the fact that point MA is in the good valley and point MB is in the bad valley, and thus the steepest descents starting from these two points should end up in different minima.

For the minimum associated with point MC, we find it approaches the minimum associated with point MA except when we start from the symmetry point MC where $w_2 = w_3$. The trajectories of the steepest descents strating from MA and MC get closer to each other the further we pursue from the origin. The fact that they approach each other only very gradually indicates that the associated valley has a very broad and flat floor further out from the origin. When we start from the symmetry point MC with $w_2 = w_3 = -0.81$, we slide down on the crest of the ridge to a point with an error value of 2.6673. This is not an actual minimum. However, since the weights at this point which is one of the symmetry planes of the error surface have the following symmetries: $w_2 = w_3$, $w_4 = w_7$, $w_5 = w_8$, and $w_6 = w_9$, the trajectory will maintain this symmetry state unless we perturb it deliberately, e.g., by eliminating one of the equalities listed above.

Table 5-1
Comparison of the Stopping Point of Different Learning Algorithms Starting from Point MA in Network 8.1.

Method	ST	C-5	SI	WD	G0.9
Error	0.0803	0.0009	0.000001	0.0019	2.6×10^{-19}
w_1	2.86	4.97	8.29	4.87	3.21
w_2	-3.00	-5.00	-8.71	-5.57	-3.39
w_3	-3.00	-5.00	-8.71	-5.57	-3.39
w_4	3.00	5.00	3.93	2.54	3.02
w_5	3.00	5.00	4.26	2.81	3.11
w_6	-3.00	-5.00	-4.26	-2.81	-3.11
w_7	3.00	5.00	3.92	2.54	3.02
w_8	-3.00	-5.00	-4.26	-2.81	-3.11
w_9	3.00	5.00	4.26	2.81	3.11

ST	Starting point (on cube with half-size 3)
C-5	Cube with half-size 5
R-18	Sphere with radius 18
WD	Weight decay
G0.9	Goal value of 0.9 or -0.9

Table 5-2
Comparison of the Stopping Point of Different Learning Algorithms Starting from Point MB in Network 8.1.

Method	ST	C-5	R-18	WD	G0.9
Error	2.0969	2.0028	2.000014	2.0028	1.62
w_1	0.00	0.00	0.00	0.00	0.00
w_2	3.00	5.00	5.16	3.48	3.70
w_3	-3.00	-5.00	-5.16	-3.48	-3.70
w_4	3.00	5.00	8.12	5.19	8.64
w_5	3.00	5.00	8.12	5.19	8.64
w_6	1.05	1.08	1.83	1.61	0.84
w_7	3.00	5.00	8.12	5.19	8.64
w_8	3.00	5.00	8.12	5.19	8.64
w_9	-1.05	-1.08	-1.83	-1.61	-0.84

ST	Starting point (on cube with half-size 3)
C-5	Cube with half-size 5
R-18	Sphere with radius 18
WD	Weight decay
G0.9	Goal value of 0.9 or -0.9

Table 5-3
Comparison of the Stopping Point of Different Learning Algorithms Starting from Point MC where $w_2 = w_3 = -0.81$ in Network 8.1.

Method	ST	C-5	R-18	WD	G0.9
Error	2.8835	2.7037	2.6673	2.6721	2.1600
w_1	0.82	1.36	3.18	2.54	2.03
w_2	-0.81	-1.04	-1.94	-1.62	-1.33
w_3	-0.81	-1.04	-1.94	-1.62	-1.33
w_4	3.00	5.00	9.65	7.40	14.72
w_5	3.00	5.00	5.48	4.30	7.72
w_6	-3.00	-5.00	-5.48	-4.30	-7.72
w_7	3.00	5.00	9.56	7.40	14.72
w_8	3.00	5.00	5.48	4.30	7.72
w_9	-3.00	-5.00	-5.48	-4.30	-7.72

ST	Starting point (on cube with half-size 3)
C-5	Cube with half-size 5
R-18	Sphere with radius 18
WD	Weight decay
G0.9	Goal value of 0.9 or -0.9

Table 5-4
Comparison of the Stopping Point of Different Learning Algorithms Starting from Point MC where w_2 = -1.62 and w_3 = 0.0 in Network 8.1.

Method	ST	C-5	R-18	WD	G0.9
Error	2.8835	0.0009	1.3×10^{-6}	0.0075	2.4×10^{-19}
w_1	0.82	4.97	8.17	4.87	3.09
w_2	-1.62	-5.00	-8.54	-5.57	-3.61
w_3	0.00	5.00	8.57	5.57	3.56
w_4	3.00	5.00	4.44	2.54	4.21
w_5	3.00	5.00	4.69	2.81	3.86
w_6	-3.00	-5.00	-4.69	-2.81	-3.86
w_7	3.00	-5.00	-3.72	-2.54	-1.77
w_8	3.00	5.00	4.09	2.81	3.06
w_9	-3.00	-5.00	-4.09	-2.81	-3.06

ST	Starting point (on cube with half-size 3)
C-5	Cube with half-size 5
R-18	Sphere with radius 18
WD	Weight decay
G0.9	Goal value of 0.9 or -0.9

Table 5-5
Comparison of the Stopping Point of Different Learning Algorithms Starting from Point MC where $w_2 = 0.0$ and $w_3 = -1.62$ in Network 8.1.

Method	ST	C-5	R-18	WD	G0.9
Error	2.8835	0.0009	1.3×10^{-6}	0.0075	2.4×10^{-19}
w_1	0.82	4.97	8.17	4.87	3.09
w_2	0.00	5.00	8.57	5.57	3.56
w_3	-1.62	-5.00	-8.54	-5.57	-3.61
w_4	3.00	-5.00	-3.72	-2.54	-1.77
w_5	3.00	5.00	4.09	2.81	3.06
w_6	-3.00	-5.00	-4.09	-2.81	-3.06
w_7	3.00	5.00	4.44	2.54	4.21
w_8	3.00	5.00	4.69	2.81	3.86
w_9	-3.00	-5.00	-4.69	-2.81	-3.86

ST	Starting point (on cube with half-size 3)
C-5	Cube with half-size 5
R-18	Sphere with radius 18
WD	Weight decay
G0.9	Goal value of 0.9 or -0.9

10. GENERAL STRUCTURES OF ERROR SURFACES ASSOCIATED WITH THE PARITY FUNCTIONS

10.1. Area around the Origin

For an error surface of a layered feed-forward network performing a boolean function, the error at the origin has a moderate bad value. Specifically, for a boolean function with g input-output patterns and ON and OFF states of 1 and -1, the error at the origin equals g. This value occurs, because at the origin, where all the weights are zero, the output of the network is zero. Therefore, the square of the difference between the actual output and the goal output is 1, and the sum of the squares of the difference is g.

For an error surface derived from the parity function, the region around the origin is very flat, because the first derivatives at the origin are zero in all directions. To show this, we begin by approximating the output of the network in a region close to the origin by

$$O(\mathrm{i}_p) = \frac{1}{2}w_1 - \frac{1}{24}w_1^3.$$

For a parity function of m inputs, the function consists of 2^m input-output patterns. Denoting i^{th} input-output pattern by (i_i, g_i), the error near the origin can be approximated by

$$Err = \sum_{i=0}^{2^m} (g_i - \frac{1}{2}w_1 - \frac{1}{24}w_1^3)^2.$$

We note that the goal output of the function is either 1 or -1, and the number of the input-output patterns with output of 1 and -1 are the same and equal 2^{n-1}. Hence, the error function becomes

$$Err = 2^{m-1}(1 - \frac{1}{2}w_1 - \frac{1}{24}w_1^3)^2 + 2^{m-1}(-1 - \frac{1}{2}w_1 - \frac{1}{24}w_1^3)^2. \qquad (24)$$

Simplifying this expression, we have

$$Err = 2^m - 2^{m-2}w_1^2 + \varepsilon(w_1^3).$$

where $\varepsilon(\,)$ is a polynomial function with a smallest degree of 3. Taking the first derivative with respect to w_1, we have

$$\frac{\partial(Err)}{\partial w_1} = -2^{m-1}w_1 + \varepsilon'(w_1^3).$$

At the origin, where $w_1 = 0$, the first derivative of the error with respect to w_1 is zero. For the first derivatives with respect to the other weights, they must also be zero because the error function (24) is a function of w_1 only.

10.2. General Structure of Error Surfaces Associated with the Parity Functions

The following characteristics appeared in all the cases we studied:

1 Symmetries exist in an error surface. Any feature of the error surface must be thought of as a class of equivalent configurations.

2 At the origin an error surface has a moderately bad error. Close to the origin the surface is quite level and then turns upward in every direction except for a few directions where shallow valleys run out from the origin. The floors of these valleys get deeper and approach zero. One must be aware that the "origin" of weight space is an artifact. In other problem where a switching function is an odd function plus a constant (which is typical when ON = 1 and OFF = 0), all the features of the error surface will be modified according to the constant term.

3 Several additional valleys originate further away from the origin and then radiate out. Some of these valley floors approach zero as they run further out.

4 There are no actual minima except at infinity. This characteristics comes from the fact that the sigmoid function used in our study only asymptotically approaches ±1 only at infinity. However, we can create minima in a bounded region by restricting the weight space to a compact set, by weight decay, by setting the the goal values to be -0.9 or 0.9, by using a hard limiter (Fig. 2) as a switching function, or by several other methods.

10.3. Effect of the Number of Hidden Neurons

The number of hidden neurons in a layered feed-forward network affects the potential and the learning efficiency of the network in performing a desired function. A network can perform an assigned function if there is a minimum on the error surface deep enough to constitute a good solution which corresponds to a set of weights that allows the network to perform an assigned function. Comparing the errors at the deepest minima on the error surfaces of network 8.1, 8.2 and 8.3 in the XOR task (Table 6), we see that the error surface of a network with more hidden neurons has a smaller error at its deepest minimum than the error surface of a network with fewer hidden neurons. In effect, a network with more hidden neurons can perform the XOR function better than a network with fewer. This statement is also true for other functions. We therefore conclude that the accuracy with which a network can approximate a given function is higher for networks with more hidden neurons.

Table 6
The Deepest Minimum on the Error Surfaces of Networks 8.1, 8.2 and 8.3 Performing the XOR Function.

Network	8.1	8.2	8.3
Number of Hidden Neurons	2	3	4
The Error of the Deepest Minimum	0.0803	0.0605	0.0005

To enable a layered feed-forward network to perform a desired function, a proper set of weights must be assigned to the network. One algorithm for finding such a set is

back–propagation [2]. This algorithm is an iterative process that, for infinitely small step size, approximates steepest descent in the weight space of a network. If the minimum found has a small enough error, the network has learned to perform the assigned function, otherwise, the network is stuck and has failed to learn.

If the starting point of the back-propagation process is chosen randomly in weight space, the chance that back-propagation will locate a particular minimum is proportional to the percent of the weight space covered by the collection zone of that minimum. That is, the success rate of back-propagation with a random starting point in training a network is proportional to the percent of the weight space covered by the collection zones of all the good minima. This success rate is a measure of how likely it is that a network will learn to perform an assigned task when using a steepest descent algorithm.

Comparing the fraction of the weight spaces covered by the collection zones of all the good minima in networks 8.1, 8.2, and 8.3 (Table 7), we see that the percentages increase from network 8.1 to 8.3. These examples indicate that the fraction of weight space covered by all the good minima is higher in a network with more hidden neurons. We therefore conclude that the learning success probability of a network can be increased by adding hidden neurons to the network.

Table 7
Percent of the Weight Space Covered by the Collection Zones of All the Good Minima in Networks 8.1, 8.2 and 8.3.

Network	8.1	8.2	8.3
Number of Hidden Neurons	2	3	4
Percent of the Weight Space Covered by the Good Collection Zone	60.3	93.9	99.3

11. METHODS FOR LOCATING GOOD MINIMA

Considering the information obtained from this study of several error surfaces, we suggest three methods for increasing the chance of finding a good minimum in a bounded weight space while restricting ourselves to gradient descent on the error surface. The success rate can be further increased by various dynamic methods, e.g., by taking large steps of varying size, by using the back-propagation with a momentum term, by adding noise to the system, or in several other ways. This issue is beyond the scope of this paper, which considers only the static nature of the error surface.

A first method exploits the fact found in the parity networks that bad valleys usually start further out from the origin, and that the valleys starting near the origin lead to good minima. In this approach, we first construct a small sphere centered at the origin. We then locate a minimum on this sphere with the method discussed in section 6.3. From there we run a steepest descent to a minimum in the weight space. If we define the radius of the searched sphere to be less than the distance between the origin and the starting point of any bad valleys, we ensure that the steepest descent starting from the minimum on the sphere will definitely lead to a good minimum.

However, we usually do not have priori knowledge about bad valleys. Also, finding the minimum on a sphere is complex and expensive to compute. We therefore recommend a simpler version of the above method. This method uses steepest descent starting from a point randomly chosen near the origin. This will significantly increase the chance of sliding first into a good valley. Starting steepest descent near the origin also decreases the chance of being trapped in bad minima whose collection zones are small and mostly concentrated in the regions near the boundaries of the weight domain. Examples of such minima are minimum MC in network 8.1, minimum MG in network 8.2, and minimum MO in network 8.3.

We used some statistical probing to verify the statement above. For each error surfaces studied earlier, we ran steepest descents from a thousand starting points randomly distributed in a cube of a given half-size. The percentages of the steepest descents stopping at each minimum in a weight domain corresponding to a cube of half-size 3.0 is given in Table B-1 through B-4 in appendix B. Comparing these data with those in Table 1, 2 and 3, we find that the steepest descents with starting points distributed in a cube of half-size 1.5 has a better chance of locating a good minimum than those with starting points distributed in a cube of half-size 3 (see Table 8). Lippeman[3] also suggests that the starting weights of back-propagation should be small.

The restriction of a starting point near the origin has one disadvantage. Since the gradient of an error function in an area around the origin is small, a steepest descent must run many iterations to get out of that area and so it slows down the convergence rate.

Table 8
The Comparison of Two Classes of Steepest Descents in Searching the Minima
A) The Starting Points of the Processes are randomly distributed in a cube of half-size 1.5.
B) The Starting Points of the Processes are randomly distributed in a cube of half-size 3.0.

Network	8.1	8.2	8.3
Percent of Trajectories in Class A Approaching good Minima	74.1	99.2	100
Percent of Trajectories in Class B Approaching good Minima	60.3	93.9	99.3

The third method is useful where the weight space is bounded by a cube. This method assumes that all minima lie close to corners of the cube, and that large gradients exist in the region further out from the origin. Exploiting these facts, we start our steepest descents near the corners of the cube. We may need to run several steepest descents before we can locate a *good* minimum. However, we will still be able to find a good minimum rather quickly, because each run will be very fast.

12. CONCLUSION

Layered feed-forward networks have recently received renewed interest because of their ability to represent complex mappings. This ability is completely described by the corresponding *error surface*. Such an error surface can be obtained by plotting the maximum error or the sum of the squares of the errors, where the individual errors are the deviation of the network's actual output from its desired goal function. The error surfaces of a network with W weights are W-dimensional surfaces embedded in a $W+1$-dimensional space and defined over a 'weight space', a W-dimensional domain given by the ranges of the W weights. The error surface can be viewed as a measure of the deviation of a desired mapping from a mapping represented by the network for all possible combinations of weight strengths. Understanding the structure of the error surface will help us answer many questions about the network's properties.

In this report, we have chosen to study a class of simple three-layer networks with complete connectivity between adjacent layers. We have found that error surfaces of these networks exhibit two kinds of symmetries. The first symmetry occurs, because the signs of all the weights connected to any hidden neuron can be inverted simultaneously without changing the behavior of the networks. The second symmetry occurs, because the connection patterns of all neurons in the hidden layer are identical; thus any permutation of these neuron roles leads to the same network behavior. When these networks perform the Exclusive-Or or the parity functions, their error surfaces have two other symmetries. The first arises from arbitrary ordering of the input neurons, and the second arises from the fact that simultaneous complementations of any two inputs do not change the definition of the functions.

Knowing these symmetries, we can define the region of a weight space such that the error surface defined over this chosen fraction possesses every aspect of the entire error surface. This region, referred to as a *representative sector*, can be located by recursively subdividing the weight space by the implied hyper-planes. This representative sector is important, because it reduces the scope of the visualization task, leaving only the region in the representative sector to be explored in order to gain full understanding of the behavior of the network.

We have studied in detail a few small networks performing the Exclusive-OR function of 2 or 3 inputs. For each case, we found that the surface at the origin is a saddle point, and the area around the origin is very level. By limiting the weight space to the region within a cube with half-size 3, we then employed steepest descent with random starting points to search for all minima on the error surface. For each minimum found, we determined its location, its error value, and the characteristics of the associated valley. We have also found that the minima found within a restricted weight space are consistent with those found by employing weight decay or by using goal values of 0.9 or -0.9 to limit the range of all the weights.

Comparing the error surfaces of the sample networks, we found that the accuracy with which a network can approximate a given function and the learning success probability are higher for a network with more hidden neurons. Finally, three methods for increasing the chance of finding a good minimum when using gradient descent emerged from the understanding of the structure of the error surface gained in this study.

REFERENCES

[1] T. J. Sejnowski and C. R. Rosenberg, "NETtalk: A Parallel Network that learns to Read Aloud," tech. report JHU/EECS-86-01, The Johns Hopkins Univ., EE and CS tech. reports, 1986.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," Parallel Distributed Processing: Explorations in the Microstructure of Cognition, eds. D.E. Rumelhart, J. L. McClelland, and the PDP Research Group, Bardford Books, Cambridge, MA, 1986.

[3] R. P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, Apr. 1987. pp 4-22.

[4] J. Denker, D. Schwartz, B. Wittner, S. Solla, J. Hopfield, R. Howard, and L. Jackel, "Automatic Learning, Rule Extraction, and Generalization," Complex Systems 1, Oct 1987, pp 877-922.

[5] N. Nilsson, "Learning Machines, Foundations of Trainable Pattern-Classifying System," McGraw-Hill, Inc. 1965.

[6] S. E. Fahlman, G. E. Hinton, "Connectionist Architectures for Artificial Intelligence," IEEE Trans, Computer, Jan 1987, pp 100-109.

[7] R. Courant, "Differential and Integral Calculus," Vol2 New York: Interscience, 1936, pp 298-304

[8] A. H. Kramer, A. Sangiovanni-Vincentelli, "Efficient Parallel Learning Algorithms for Neural Networks," [To be published in IEEE Conference on Neural Information on Processing Systems, Natural and Synthetic.]

APPENDIX A Symmetry Operations

Table A-1
Symmetry Operations Associated with the Error Surface
of
Network 8.1 Performing the XOR mapping

Notation	Description
opH1	w2 → -w2, w4 → -w4, w5 → -w5, w6 → -w6
opH2	w3 → -w3, w7 → -w7, w8 → -w8, w9 → -w9,
opHx21	w2 ⟷ w3, w4 ⟷ w7, w5 ⟷ w8, w6 ⟷ w9,
opIx	w5 ⟷ w6, w8 ⟷ w9.
opI1	w5 ⟷ -w5, w8 ⟷ -w8, w1 ⟷ -w1, w2 ⟷ -w2, w3 ⟷ -w3.
opI2	w6 ⟷ -w6, w9 ⟷ -w9, w1 ⟷ -w1, w2 ⟷ -w2, w3 ⟷ -w3.
Number of Equivalent Configurations 64	
Representative Sector $w4 \geq w7 \geq 0; \quad w5 \geq w6 \geq 0$	

Table A-2
Symmetry Operations Associated with the Error Surface
of
Network 8.2 Performing the XOR mapping

Notation	Description
opH1	w2 → -w2, w5 → -w5, w6 → -w6, w7 → -w7
opH2	w3 → -w3, w8 → -w8, w9 → -w9, w10 → -w10
opH3	w4 → -w4, w11 → -w11, w12 → -w12, w13 → -w13
opHx132	w3 ⟷ w4, w8 ⟷ w11, w9 ⟷ w12, w10 ⟷ w13
opHx213	w3 ⟷ w4, w8 ⟷ w11, w9 ⟷ w12, w10 ⟷ w13
opHx231	w3 → w2, w4 → w3, w2 → w4, w8 → w5, w9 → w6, w10 → w7, w11 → w8, w12 → w9, w13 → w10, w5 → w11, w6 → w12, w7 → w13
opHx312	w4 → w2, w2 → w3, w3 → w4, w11 → w5, w12 → w6, w13 → w7, w5 → w8, w6 → w9, w7 → w10, w8 → w11, w9 → w12, w10 → w13
opHx321	w2 ⟷ w4, w5 ⟷ w11, w6 ⟷ w12, w7 ⟷ w13
opIx	w6 ⟷ w7, w9 ⟷ w10, w12 ⟷ w13
opI1	w1 → -w1, w2 → -w2, w3 → -w3, w4 → -w4, w6 → -w6, w9 → -w9, w12 → -w12
opI2	w1 → -w1, w2 → -w2, w3 → -w3, w4 → -w4, w7 → -w7, w10 → -w10, w13 → -w13
Number of Equivalent Configurations 384	
Representative Sector $w5 \geq w8 \geq w11 \geq 0; \quad w6 \geq w7 \geq 0$	

Table A-3
Symmetry Operations Associated with the Error Surface
of
Network 8.3 Performing the XOR mapping

Notation	Description
opH1	w2 → -w2, w6 → -w6, w7 → -w7, w8 → -w8
opH2	w3 → -w3, w9 → -w9, w10 → -w10, w11 → -w11
opH3	w4 → -w4, w12 → -w12, w13 → -w13, w14 → -w14
opH4	w5 → -w5, w15 → -w15, w16 → -w16, w17 → -w17
opHx1243	w4 ⟷ w5, w12 ⟷ w15, w13 ⟷ w16, w14 ⟷ w17
opHx1324	w3 ⟷ w4, w9 ⟷ w12, w10 ⟷ w13, w11 ⟷ w14
opHx1342	w4 → w3, w5 → w4, w3 → w5, w12 → w9, w13 → w10, w14 → w11, w15 → w12, w16 → w13, w17 → w14, w9 → w15, w10 → w16, w11 → w17
opHx1423	w5 → w3, w3 → w4, w4 → w5, w15 → w9, w16 → w10, w17 → w11, w9 → w12, w10 → w13, w11 → w14, w12 → w15, w13 → w16, w14 → w17
opHx1432	w3 ⟷ w5, w9 ⟷ w15, w10 ⟷ w16, w11 ⟷ w17
opHx2134	w2 ⟷ w3, w6 ⟷ w9, w7 ⟷ w10, w8 ⟷ w11
opHx2143	w3 → w2, w2 → w3, w5 → w4, w4 → w5, w9 → w6, w10 → w7, w11 → w8, w6 → w9, w7 → w10, w8 → w11, w15 → w12, w16 → w13, w17 → w14, w12 → w15, w13 → w16, w14 → w17
opHx2314	w3 → w2, w4 → w3, w2 → w4, w9 → w6, w10 → w7, w11 → w8, w12 → w9, w13 → w10, w14 → w11, w6 → w12, w7 → w13, w8 → w14
opHx2341	w3 → w2, w4 → w3, w5 → w4, w2 → w5, w9 → w6, w10 → w7, w11 → w8, w12 → w9, w13 → w10, w14 → w11, w15 → w12, w16 → w13, w17 → w14, w6 → w15, w7 → w16, w8 → w17
opHx2413	w3 → w2, w5 → w3, w2 → w4, w4 → w5, w9 → w6, w10 → w7, w11 → w8, w15 → w9, w16 → w10, w17 → w11, w6 → w12, w7 → w13, w8 → w14, w12 → w15, w13 → w16, w14 → w17
opHx2431	w3 → w2, w5 → w3, w2 → w5, w9 → w6, w10 → w7, w11 → w8, w15 → w9, w16 → w10, w17 → w11, w6 → w15, w7 → w16, w8 → w17
opHx3124	w4 → w2, w2 → w3, w3 → w4, w12 → w6, w13 → w7, w14 → w8, w6 → w9, w7 → w10, w8 → w11, w9 → w12, w10 → w13, w11 → w14
opHx3142	w4 → w2, w2 → w3, w5 → w4, w3 → w5, w12 → w6, w13 → w7, w14 → w8, w6 → w9, w7 → w10, w8 → w11, w15 → w12, w16 → w13, w17 → w14, w9 → w15, w10 → w16, w11 → w17
opHx3214	w2 ⟷ w4, w6 ⟷ w12, w7 ⟷ w13, w8 ⟷ w14
opHx3241	w4 → w2, w5 → w4, w2 → w5, w12 → w6, w13 → w7, w14 → w8, w15 → w12, w16 → w13, w17 → w14, w6 → w15, w7 → w16, w8 → w17
opHx3412	w4 → w2, w5 → w3, w2 → w4, w3 → w5, w12 → w6, w13 → w7, w14 → w8, w15 → w9, w16 → w10, w17 → w11, w6 → w12, w7 → w13, w8 → w14, w9 → w15, w10 → w16, w11 → w17
opHx3421	w4 → w2, w5 → w3, w3 → w4, w2 → w5, w12 → w6, w13 → w7, w14 → w8, w15 → w9, w16 → w10, w17 → w11, w9 → w12, w10 → w13, w11 → w14, w6 → w15, w7 → w16, w8 → w17

Table A-3 Continued
Symmetry Operations Associated with the Error Surface
of
Network 8.3 Performing the XOR mapping

Notation	Description
opHx4123	w5 → w2, w2 → w3, w3 → w4, w4 → w5, w15 → w6, w16 → w7, w17 → w8, w6 → w9, w7 → w10, w8 → w11, w9 → w12, w10 → w13, w11 → w14, w12 → w15, w13 → w16, w14 → w17
opHx4132	w5 → w2, w2 → w3, w3 → w5, w15 → w6, w16 → w7, w17 → w8, w6 → w9, w7 → w10, w8 → w11, w9 → w15, w10 → w16, w11 → w17
opHx4213	w5 → w2, w2 → w4, w4 → w5, w15 → w6, w16 → w7, w17 → w8, w6 → w12, w7 → w13, w8 → w14, w12 → w15, w13 → w16, w14 → w17
opHx4231	w2 ←→ w5, w6 ←→ w15, w7 ←→ w16, w8 ←→ w17
opHx4312	w5 → w2, w4 → w3, w2 → w4, w3 → w5, w15 → w6, w16 → w7, w17 → w8, w12 → w9, w13 → w10, w14 → w11, w6 → w12, w7 → w13, w8 → w14, w9 → w15, w10 → w16, w11 → w17
opHx4321	w5 → w2, w4 → w3, w3 → w4, w2 → w5, w15 → w6, w16 → w7, w17 → w8, w12 → w9, w13 → w10, w14 → w11, w9 → w12, w10 → w13, w11 → w14, w6 → w15, w7 → w16, w8 → w17
opIx	w7 ←→ w8, w10 ←→ w11, w13 ←→ w14, w16 ←→ w17
opI1	w1 → -w1, w2 → -w2, w3 → -w3, w4 → -w4, w5 → -w5, w7 → -w7, w10 → -w10, w13 → -w13, w16 → -w16
opI2	w1 → -w1, w2 → -w2, w3 → -w3, w4 → -w4, w5 → -w5, w8 → -w8, w11 → -w11, w14 → -w14, w17 → -w17

Number of Equivalent Configurations 3072

Representative Sector

$w6 \geq w9 \geq w12 \geq w15 \geq 0; \quad w7 \geq w8 \geq 0$

Table A-4
Symmetry Operations Associated with the Error Surface
of
Network 8.4 Performing the 3-input Parity Mapping

Notation	Description
opH1	w2 → -w2, w5 → -w5, w6 → -w6, w7 → -w7, w8 → -w8
opH2	w3 → -w3, w9 → -w9, w10 → -w10, w11 → -w11, w12 → -w12
opH3	w4 → -w4, w13 → -w13, w14 → -w14, w15 → -w15, w16 → -w16
opHx132	w3 ⟷ w4, w9 ⟷ w13, w10 ⟷ w14, w11 ⟷ w15, w12 ⟷ w16
opHx213	w2 ⟷ w3, w5 ⟷ w9, w6 ⟷ w10, w7 ⟷ w11, w8 ⟷ w12
opHx231	w3 → w2, w4 → w3, w2 → w4, w9 → w5, w10 → w6, w11 → w7, w12 → w8, w13 → w9, w14 → w10, w15 → w11, w16 → w12, w5 → w13, w6 → w14, w7 → w15, w8 → w16
opHx312	w4 → w2, w2 → w3, w3 → w4, w13 → w5, w14 → w6, w15 → w7, w16 → w8, w5 → w9, w6 → w10, w7 → w11, w8 → w12, w9 → w13, w10 → w14, w11 → w15, w12 → w16
opHx321	w2 ⟷ w4, w5 ⟷ w13, w6 ⟷ w14, w7 ⟷ w15, w8 ⟷ w16
opIx132	w7 ⟷ w8, w11 ⟷ w12, w15 ⟷ w16
opIx213	w6 ⟷ w7, w10 ⟷ w11, w14 ⟷ w15
opIx231	w7 → w6, w8 → w7, w6 → w8, w11 → w10, w12 → w11, w10 → w12, w15 → w14, w16 → w15, w14 → w16,
opIx312	w8 → w6, w6 → w7, w7 → w8, w12 → w10, w10 → w11, w11 → w12, w16 → w14, w14 → w15, w15 → w16,
opIx321	w6 ⟷ w8, w10 ⟷ w12, w14 ⟷ w16
opI1	w1 → -w1, w2 → -w2, w3 → -w3, w4 → -w4 w6 → -w6, w10 → -w10, w14 → -w14
opI2	w1 → -w1, w2 → -w2, w3 → -w3, w4 → -w4 w7 → -w7, w11 → -w11, w15 → -w15
opI3	w1 → -w1, w2 → -w2, w3 → -w3, w4 → -w4 w8 → -w8, w12 → -w12, w16 → -w16

Number of Equivalent Configurations 2304

Representative Sector

$w5 \geq w9 \geq w13 \geq 0; \quad w6 \geq w7 \geq w8 \geq 0$

APPENDIX B Influence of Starting Point on Gradient Descent

Table B-1
The Comparison of Two Classes of Steepest Descents in Searching the Minima in the Error Surface of Network 8.1 Performing the XOR Mapping.
A) The Starting Points of the Processes are randomly distributed in $[-1.5, 1.5]^9$
B) The Starting Points of the Processes are randomly distributed in $[-3.0, 3.0]^9$

Identifier	MA	MB	MC
Error	0.0803	2.0969	2.8835
Percent of Trajectories in Class A Approaching the Point	74.1	25.6	0.3
Percent of Trajectories in Class B Approaching the Point	60.3	37.5	2.2

Table B-2
The Comparison of Two Classes of Steepest Descents in Searching the Minima in the Error Surface of Network 8.2 Performing the XOR Mapping.
A) The Starting Points of the Processes are randomly distributed in $[-1.5, 1.5]^{13}$
B) The Starting Points of the Processes are randomly distributed in $[-3.0, 3.0]^{13}$

Identifier	MD	ME	MF	MG
Error	0.0605	0.0665	0.0803	2.0965
Percent of Trajectories in Class A Approaching the Point	74.8	20.7	3.7	0.8
Percent of Trajectories in Class B Approaching the Point	67.1	25.3	1.5	6.1