Communicated by Shun-ichi Amari and Halbert White

On the Geometry of Feedforward Neural Network Error Surfaces

An Mei Chen
Haw-minn Lu
University of California, San Diego, CA USA

Robert Hecht-Nielsen
HNC, Inc.
and
University of California, San Diego, CA USA

Many feedforward neural network architectures have the property that their overall input–output function is unchanged by certain weight permutations and sign flips. In this paper, the geometric structure of these *equioutput* weight space transformations is explored for the case of multilayer perceptron networks with *tanh* activation functions (similar results hold for many other types of neural networks). It is shown that these transformations form an algebraic group isomorphic to a direct product of Weyl groups. Results concerning the root spaces of the Lie algebras associated with these Weyl groups are then used to derive sets of simple equations for minimal sufficient search sets in weight space. These sets, which take the geometric forms of a wedge and a cone, occupy only a minute fraction of the volume of weight space. A separate analysis shows that large numbers of copies of a network performance function optimum weight vector are created by the action of the equioutput transformation group and that these copies all lie on the same sphere. Some implications of these results for learning are discussed.

1 Introduction

For the sake of concreteness, we will concentrate in this paper on the "multilayer perceptron" or "backpropagation" feedforward neural network architecture (Rumelhart *et al.* 1986; Hecht-Nielsen 1992). However, many of the results we present can be reformulated to apply to other neural network architectures as well [e.g., the radial basis function networks of Reilly *et al.* (1982), Broomhead and Lowe (1988), Moody and Darken (1989), and Poggio and Girosi (1990); the ART networks of Carpenter and Grossberg (1991); counterpropagation networks (Hecht-Nielsen

1991); and the mutual information preserving networks of Linsker (1988) and Becker and Hinton (1992)].

The layers of the neural networks we consider in this paper are assumed to have units with transfer functions of the form

$$\begin{aligned} z_{li} &= s(I_{li}) \\ I_{li} &= \sum_{j=0}^{M_l} w_{lij} z_{(l-1)i} \end{aligned}$$

for $l > 1$, where

$$\begin{aligned} s(u) &= \tanh(u) \text{ for layers 2 through } K-1 \\ &= u \text{ for layer } K \\ K &= \text{number of layers in the network} \\ &\quad \text{(including input and output layers)} \\ l &= \text{layer number (1 through } K\text{)} \\ M_l &= \text{number of units on layer } l\text{, assumed to be} > 1 \\ z_{1i} &= x_i = i\text{th component of the} \\ &\quad \text{external input vector } \mathbf{x} \\ &\quad 1 \le i \le n \\ z_{Kj} &= y'_j = j\text{th component of the} \\ &\quad \text{network output vector } \mathbf{y}' \\ &\quad 1 \le j \le m \\ z_{l0} &\equiv 1.0 \\ w_{lij} &= \text{weight of unit } i \\ &\quad \text{of layer } l \text{ associated with} \\ &\quad \text{input } z_{(l-1)j} \text{ from layer } l-1 \end{aligned}$$

Each layer in this architecture receives inputs from all of the units of the previous layer, but none from any other layer. (Note: all of our results either remain the same or easily generalize when connections skip layers, but the mathematical notation becomes messy. Thus, only the simplest case is presented here.)

The *network weight vector* of a multilayer perceptron neural network is the q-dimensional real Euclidean vector $\mathbf{w}$ whose components consist of all of the weights of the network in some fixed order. We shall refer to the space of all such weight vectors (namely, R^q) as *weight space* and denote it by W. Clearly, the network weight vector determines the input–output transfer function of the network.

For the purposes of this paper, we shall assume that each multilayer perceptron neural network under consideration is being used to approximate a fixed, square integrable (i.e., L_2) function

$$f : A \subset R^n \longrightarrow R^m$$

where A is a compact subset of R^n. Further, we shall assume that the performance of the network is being measured by some *performance function* $F(\mathbf{w})$ that depends only on the network's input–output transfer function (which depends only on the selection of the network weight vector $\mathbf{w}$) and on the manner in which the $\mathbf{x}$ vectors in A are chosen (which is assumed to be in accordance with a fixed scheme: namely, random selection with respect to a fixed smooth [C^∞ probability density function $\rho(\mathbf{x})$ such that the elements of a random vector with this density are linearly independent with probability one]. It suffices (but is not necessary) that the covariance matrix associated with ρ exist and be finite and nonsingular. Note that this method of choosing the input vectors ensures that they will not have a fixed linear relationship with one another—which could introduce undesired symmetries into the weights of the first hidden layer.

Given network performance function $F(\mathbf{w})$, we can view F as a surface hovering over the weight space W with its altitude at each point $\mathbf{w}$ determined by the performance of the network with that selection of weight values. Given a multilayer perceptron network, a function $f : A \subset R^n \longrightarrow R^m$ to approximate, a probability density function $\rho(\mathbf{x})$, and a particular performance function $F(\mathbf{w})$, we shall refer to such a surface as the *performance surface* of the network.

These assumptions about the network performance function are very mild. For example, functions as diverse as mean squared error, median of squared errors, and the supremum of errors, namely

$$\begin{aligned}
F_2(\mathbf{w}) &= \int_A |f(\mathbf{x}) - \mathbf{y}'(\mathbf{x},\mathbf{w})|^2 \;\rho(\mathbf{x})\; d\mathbf{x} \\
&= \lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} |f(\mathbf{x}_k) - \mathbf{y}'(\mathbf{x}_k,\mathbf{w})|^2 \\
F_{\mathrm{m}}(\mathbf{w}) &= \lim_{N\to\infty} \mathrm{median}[|f(\mathbf{x}_1) - \mathbf{y}'(\mathbf{x}_1,\mathbf{w})|^2, \ldots, |f(\mathbf{x}_N) - \mathbf{y}'(\mathbf{x}_N,\mathbf{w})|^2] \\
F_\infty(\mathbf{w}) &= \sup_{\mathbf{x}\in A \text{ and } \rho(\mathbf{x})>0} |f(\mathbf{x}) - \mathbf{y}'(\mathbf{x},\mathbf{w})|
\end{aligned}$$

are accommodated within the definition, where $\mathbf{y}'(\mathbf{x},\mathbf{w})$ is the output of the multilayer perceptron network, which is an approximation of the desired output of the network $\mathbf{y} = f(\mathbf{x})$.

The key element of the performance function definition is its dependence only on the input–output transfer function of the network. This allows the network performance to be evaluated not only in terms of just the errors it makes, but also, if desired, in terms of other factors—such as the curvature of its "approximating surface" (as determined by functions of derivatives of $\mathbf{y}'(\mathbf{x}, \mathbf{w})$ with respect to the components of the input vector $\mathbf{x}$), as in the networks of Bishop (1991, 1990) and Poggio and Girosi (1990). However, explicit dependence of the performance function on factors that are not determined by the input–output behavior of the network—such as direct dependence on the number of hidden units—is not allowed by this definition.

The main focus of this paper is the study of geometric transformations of weight space that have the property that they leave the input–output transformation of the neural network unchanged. Obviously, such transformations will also leave all network performance functions unchanged. We begin in Section 2 by showing that all such *equioutput* transformations are compositions of two simple classes of isometries. Following this, we show that the set of all equioutput transformations forms an algebraic group of surprisingly large order.

The fact that there exists a large group of equioutput transformations in weight space implies that performance surfaces are highly redundant, since each network weight vector is *equivalent* (in terms of determining the same network input–output transfer function and performance) to a multitude of other weight vectors. Thus, if we are searching for an optimum of a performance function, it would seem to be possible, at least in principle, to make our search more efficient by confining it to a small subset of weight space in which all redundancy has been eliminated.

In Section 3 we proceed to further analyze our equioutput transformation group by showing that it is isomorphic to a direct product of Weyl groups. In Section 4 we then exploit known facts about these Weyl groups and the root spaces of their associated Lie algebras to derive a set of simple inequalities that defines nonredundant search sets having the geometric forms of a wedge and a cone. These *minimal sufficient search sets* occupy only a minute fraction of the volume of weight space and contain no two equivalent weight vectors, while containing weight vectors equivalent to every weight vector in the space.

In Section 5 we consider yet another implication of our transformation group: that each weight vector that optimizes the network performance function is equivalent to many other such optima, and that all of these lie on the same sphere.

Finally, in Section 6 we consider the implications of the results of Sections 2, 3, 4, and 5 for neural network learning.

2 Equioutput Transformations

We begin by studying the properties of weight space transformations that leave the network input–output transfer function unchanged. These transformations are now defined.

Definition 1. An *equioutput transformation* is an analytic (i.e., continuous and expandable in a power series around any point) mapping $g : W \longrightarrow W$ from weight space to weight space which leaves the output of the neural network unchanged. In other words,

$$\mathbf{y}'(\mathbf{x}, g(\mathbf{w})) = \mathbf{y}'(\mathbf{x}, \mathbf{w})$$

for all $\mathbf{x} \in R^n$ and all $\mathbf{w} \in W$.

First, consider two types of equioutput transformations: hidden unit weight interchanges and hidden unit weight sign flips. For simplicity, we will refer to these transformations as *interchanges* and *sign flips*. An interchange consists of a network weight vector component permutation in which the weight vectors of two hidden units on the same hidden layer are simply interchanged without changing the orderings of the weights within the units. (Note: the term *unit weight vector* refers to the vector with components equal to the weights within a single unit.) A compensatory interchange of the weights of the next layer units that receive the inputs from the two interchanged units then removes the effect on the network output of the exchange of weights in the previous layer (see Fig. 1).

The other type of equioutput transformation is where the weight vector of a hidden layer unit is multiplied by -1 (resulting in a sign flip of the output of the unit—since *tanh* is an odd function). A compensatory sign flip is then carried out on all of the weights of units of the next layer associated with the input from the sign flipped unit output (see Fig. 2). We now show that:

Theorem 1. *All equioutput transformations of W to W are compositions of interchange and sign flip transformations.*

Proof. By induction. Let g be any equioutput transformation. We first note that, since $\mathbf{y}'$ is the output of the network for input $\mathbf{x}$, we have

$$\begin{aligned} \mathbf{y}'_i(\mathbf{x}, \mathbf{w}) &= \sum_{j=0}^{M_{K-1}} w_{Kij}\, z_{(K-1)j}(\mathbf{x}, \mathbf{w}) \\ &= \sum_{j=0}^{M_{K-1}} g(\mathbf{w})_{Kij}\, z_{(K-1)j}(\mathbf{x}, g(\mathbf{w})) \\ &= \mathbf{y}'_i(\mathbf{x}, g(\mathbf{w})) \end{aligned} \tag{2.1}$$

where $\mathbf{y}'_i(\mathbf{x}, \mathbf{w})$ is the output of the original network, $\mathbf{y}'_i(\mathbf{x}, g(\mathbf{w}))$ is the output of the network with weight vector $g(\mathbf{w})$ (the *transformed* network), and where $z_{(K-1)j}(\mathbf{x}, \mathbf{w})$ and $z_{(K-1)j}(\mathbf{x}, g(\mathbf{w}))$ are, respectively, the outputs of the jth units of the last hidden layers of the original and transformed networks.

The first step is to take the second partial derivatives of both sums of equations 2.1 with respect to output layer weights w_{Kij} and w_{Kuv}. Note

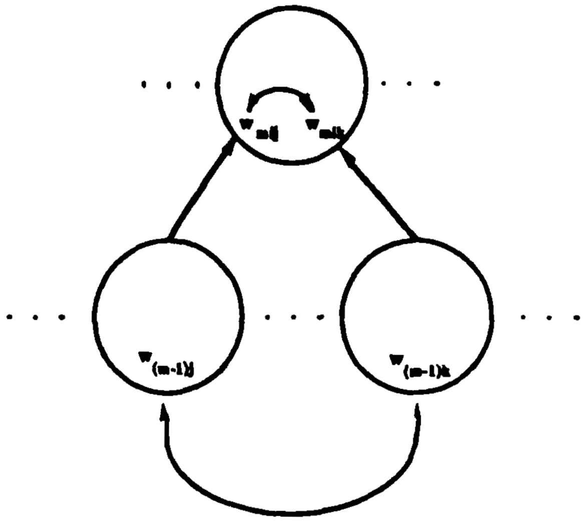

Figure 1: *Interchange* transformations involve interchanging the weight vectors of two units on a hidden layer (which has the same effect as interchanging the units themselves). The weights within the units of the next layer that act upon the inputs from the interchanged units are themselves interchanged. The net result is no change in the outputs of these next-layer units.

that the first partial of the first sum with respect to w_{Kij} is equal to $z_{(K-1)j}(\mathbf{x},\mathbf{w})$ and the second partial of this sum is zero because the output of the last hidden layer of the original network has no dependence on the output layer weights. Thus, we get

$$\begin{aligned} 0 \ = \ & \sum_{l=0}^{M_{K-1}} \frac{\partial^2 g(\mathbf{w})_{Kil}}{\partial w_{Kuv}\,\partial w_{Kij}}\, z_{(K-1)l}(\mathbf{x}, g(\mathbf{w})) \\ & + \sum_{l=0}^{M_{K-1}} \frac{\partial g(\mathbf{w})_{Kil}}{\partial w_{Kij}} \frac{\partial z_{(K-1)l}(\mathbf{x}, g(\mathbf{w}))}{\partial w_{Kuv}} \\ & + \sum_{l=0}^{M_{K-1}} \frac{\partial g(\mathbf{w})_{Kil}}{\partial w_{Kuv}} \frac{\partial z_{(K-1)l}(\mathbf{x}, g(\mathbf{w}))}{\partial w_{Kij}} \\ & + \sum_{l=0}^{M_{K-1}} g(\mathbf{w})_{Kil} \frac{\partial^2 z_{(K-1)l}(\mathbf{x}, g(\mathbf{w}))}{\partial w_{Kuv}\,\partial w_{Kij}} \end{aligned} \tag{2.2}$$

for the second partial derivative of the second sum.

If we write out the mathematical forms of the four sums of equation 2.2 we see that each nonzero term in each is a transcendental form

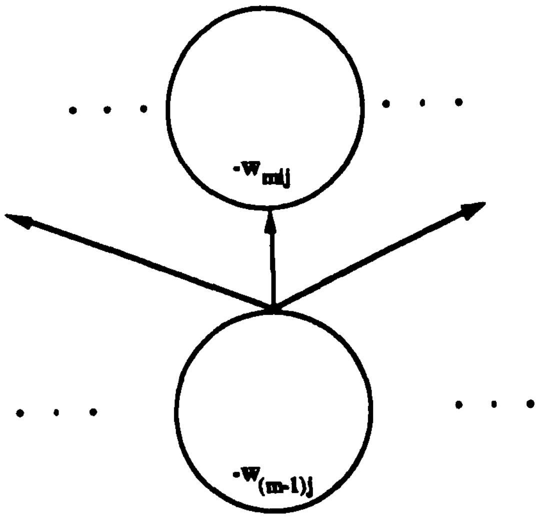

Figure 2: In a *sign flip* transformation, the signs of all of the weights of a single unit on a hidden layer are changed. The signs of the weights acting on this unit's output in all of the units on the next layer are also flipped. Again, as with interchange transformations, the outputs of the next-layer units are unchanged

that will be, by inspection, in general (i.e., for all but a set of weights of measure zero), linearly independent with respect to all of the other terms as the network input $\mathbf{x}$ is varied (except for pairs of terms in the middle two sums that have matching values of l). Thus, except for these pairs, each term in these sums must itself be identically zero. The bad sets of weights need not concern us because we can analytically continue into them. Consider the fourth sum first. In this sum the $g(\mathbf{w})_{Kil}$ are not zero in general so we must have

$$\frac{\partial^2 z_{(K-1)l}(\mathbf{x}, g(\mathbf{w}))}{\partial w_{Kuv}\, \partial w_{Kij}} = 0$$

for all u, v, i, j, and l. Therefore,

$$z_{(K-1)l}(\mathbf{x}, g(\mathbf{w})) = a_l(\mathbf{x}, \hat{\mathbf{w}}) + \sum_{i=0}^{m} \sum_{j=0}^{M_{K-1}} b_{lij}(\mathbf{x}, \hat{\mathbf{w}})\, w_{Kij} \tag{2.3}$$

where $\hat{\mathbf{w}}$ is the vector $\mathbf{w}$ with all of the output layer components w_{Kij} removed. In the first sum of equation 2.2 the $z_{(K-1)l}(\mathbf{x}, g(\mathbf{w}))$s are clearly nonzero in general, so we must have

$$\frac{\partial^2 g(\mathbf{w})_{Kil}}{\partial w_{Kuv}\,\partial w_{Kij}} = 0$$

for all u, v, i, j, and l (we will see below why this is true). Finally, we note that not all of the $\partial g(\mathbf{w})_{Kil}/\partial w_{Kuv}$ can be zero, for if they were, then the output weight $g(\mathbf{w})_{Kil}$ would not depend on any of the output weights of the original network. This cannot be, since then there would be no way to set all of the outputs of the transformed network to zero for all $\mathbf{x}$ inputs (which we can do by simply setting all of the output weights of the original network to zero). Thus, we conclude that $\partial z_{(K-1)l}(\mathbf{x}, g(\mathbf{w}))/\partial w_{Kij}$ must be zero. Thus, the weights of the hidden layers of the network do not depend upon the output layer weights [i.e., all of the $b_{lij}(\mathbf{x}, \hat{\mathbf{w}})$ of equation 2.3 must be zero].

We now explore the relationship between the output layer weights w_{Kij} and $g(\mathbf{w})_{Kij}$ of the original and transformed networks. To do this we expand both $z_{(K-1)j}(\mathbf{x}, \mathbf{w})$ and $z_{(K-1)j}(\mathbf{x}, g(\mathbf{w}))$ as power series in $\mathbf{x}$ and substitute these into equations 2.1. These expansions are given by

$$z_{(K-1)j}(\mathbf{x}, \mathbf{w}) = a_{1j} + \mathbf{x}^{\mathrm{T}}\,\mathbf{b}_{1j} + \frac{1}{2}\mathbf{x}^{\mathrm{T}}C_{1j}\mathbf{x} + \cdots$$

and

$$z_{(K-1)j}(\mathbf{x}, g(\mathbf{w})) = a_{2j} + \mathbf{x}^{\mathrm{T}}\,\mathbf{b}_{2j} + \frac{1}{2}\mathbf{x}^{\mathrm{T}}\,C_{2j}\,\mathbf{x} + \cdots$$

When we substitute these quantities into the sums of equations 2.1 we note that, since these sums are equal for all values of $\mathbf{x}$, that all of the coefficients in these power series expansions must be equal as well. Thus, we get an infinite set of linear equations

$$\begin{aligned}
\sum_{j=0}^{M_{K-1}} w_{Kij}\,a_{1j} &= \sum_{j=0}^{M_{K-1}} g(\mathbf{w})_{Kij}\,a_{2j} \\
\sum_{j=0}^{M_{K-1}} w_{Kij}\,\mathbf{b}_{1j} &= \sum_{j=0}^{M_{K-1}} g(\mathbf{w})_{Kij}\,\mathbf{b}_{2j} \\
\sum_{j=0}^{M_{K-1}} w_{Kij}\,C_{1j} &= \sum_{j=0}^{M_{K-1}} g(\mathbf{w})_{Kij}\,C_{2j} \\
&\vdots
\end{aligned}$$

and so on. Since the coefficients in the multidimensional Taylor's series for $z_{(K-1)j}(\mathbf{x}, \mathbf{w})$ and $z_{(K-1)j}(\mathbf{x}, g(\mathbf{w}))$ are set by controlling the nonoutput layer weights of the network, and since the functional forms achievable by setting these nonoutput layer weights are a rich set of functions [see Sussmann (1992) for a discussion of this property], these equations are, in general, not linearly dependent [note that as the coefficients are changed the w_{Kij}s and $g(\mathbf{w})_{Kij}$s remain fixed]. Thus, the w_{Kij}s and $g(\mathbf{w})_{Kij}$s must be linearly dependent. So, we can write each $g(\mathbf{w})_{Kij}$ as a linear combination of the w_{Kij}s, and vice versa, with coefficients that themselves cannot be functions of the output layer weights. Thus, we can write

$$g(\mathbf{w})_{Kij} = \sum_{l=0}^{M_{K-1}} d(\hat{\mathbf{w}})_{ijl}\; w_{Kil} \tag{2.4}$$

Substituting equation 2.4 into equation 2.1 and taking the partial derivative with respect to w_{Kij} then gives

$$\begin{aligned} \frac{\partial}{\partial w_{Kij}}\, \mathbf{y}_i'(\mathbf{x},\mathbf{w}) &= \frac{\partial}{\partial w_{Kij}} \sum_{u=0}^{M_{K-1}} w_{Kiu}\, z_{(K-1)u}(\mathbf{x},\mathbf{w}) = \\ z_{(K-1)j}(\mathbf{x},\mathbf{w}) &= \frac{\partial}{\partial w_{Kij}} \sum_{u=0}^{M_{K-1}} \left[\sum_{v=0}^{M_{K-1}} d(\hat{\mathbf{w}})_{iuv} w_{Kiv}\right] z_{(K-1)u}(\mathbf{x},g(\mathbf{w})) \\ &= \sum_{u=0}^{M_{K-1}} d(\hat{\mathbf{w}})_{iuj}\, z_{(K-1)u}(\mathbf{x},g(\mathbf{w})) \end{aligned} \tag{2.5}$$

Note that if we set all of the output weights but w_{Kij} to zero (without changing any of the other weights in the network), equations 2.4 and 2.5 imply that exactly one of the $d(\hat{\mathbf{w}})_{iuj}$ must equal either $+1$ or -1 for each fixed i and j. The others must be zero. This follows because neither the $d(\hat{\mathbf{w}})_{iuj}$s nor the $z_{(K-1)u}[\mathbf{x},g(\mathbf{w})]$s can be functions of the output weights.

Finally, if we substitute equations 2.4 and 2.5 into equation 2.1 we get

$$\begin{aligned} \mathbf{y}_k'(\mathbf{x},\mathbf{w}) &= \sum_{j=0}^{M_{K-1}} w_{Kkj} \left[\sum_{u=0}^{M_{K-1}} d(\hat{\mathbf{w}})_{iuj}\, z_{(K-1)u}(\mathbf{x},g(\mathbf{w}))\right] \\ &= \sum_{j=0}^{M_{K-1}} \sum_{u=0}^{M_{K-1}} d(\hat{\mathbf{w}})_{iuj}\, w_{Kkj}\, z_{(K-1)u}(\mathbf{x},g(\mathbf{w})) \\ &= \sum_{j=0}^{M_{K-1}} \left[\sum_{l=0}^{M_{K-1}} d(\hat{\mathbf{w}})_{kjl} w_{Kkl}\right] z_{(K-1)j}(\mathbf{x},g(\mathbf{w})) \\ &= \sum_{j=0}^{M_{K-1}} \sum_{u=0}^{M_{K-1}} d(\hat{\mathbf{w}})_{kuj}\, w_{Kkj}\, z_{(K-1)u}(\mathbf{x},g(\mathbf{w})) \end{aligned}$$

From this we see that $d(\hat{\mathbf{w}})_{iuj} = d(\hat{\mathbf{w}})_{kuj}$ for all i and k. Thus, the d values for each output unit are the same. Thus, the only possible equioutput transformations are those that have the effect of sign flips and interchanges on the last hidden layer. That this is true for all of the other hidden layers (if any) is clear, since we can simply "clip off" the output layer of the network and invert the *tanh* functions of the last hidden layer outputs to create another multilayer perceptron. Applying the above argument to this network then shows that all equioutput transformations act as compositions of interchanges and sign flips on the second to last hidden layer, and so on. □

We believe, but cannot prove, that the above theorem would hold even if we only demanded continuity (and not analyticity) of our equioutput transformations. In addition to our analytic equioutput transformations, there exist discontinuous *conditional equioutput transformations* as well. For example, if all of the weights acting on the output of a hidden unit are zero, then the weights of that unit can take on any values without altering the output of the network. So, this unit's transformed weights might, for example, be set to constant values under this condition, yielding a discontinuous equioutput transformation. Sussmann (1992) has studied and cataloged some of these situations (he has also established a result closely related to Theorem 1). These discontinuous transformations may be worthy of further investigation, as they establish the existence of affine subspace "generators" at at least some points on performance surfaces. If it could be shown that all performance surfaces are generated (like the cylinder $(x^2/a^2)+(y^2/a^2) = 1$ or the hyperboloid $(x^2/a^2)+(y^2/a^2)-(z^2/b^2) = 1$ in three-dimensional space can be generated by moving a line along a trajectory and correctly controlling the attitude of the line as it moves), this might provide a new line of attack for understanding the geometric structure of such surfaces. Whether performance surfaces will, in general, turn out to be generated is unclear.

3 The Structure of Group G

In this section we show that the set of all equioutput transformations forms a group. Then we analyze the structure of this group.

Theorem 2. *The set of all equioutput transformations on W forms a non-Abelian group G of order #G, with*

$$\#G = \prod_{l=2}^{K-1} (M_l!)(2^{M_l})$$

Proof. We first note that the set of interchange transformations involving the interchange of unit weight vectors on hidden layer l is in one-to-one correspondence with the set of all permutations of M_l elements. Thus, there are $(M_l!)$ different interchange transformations for layer l. The number of sign flips is the same as the number of binary numbers with M_l bits, or 2^{M_l}. It is easy to show that the interchange and sign flip transformations of one layer commute with those of any other layer. Thus, they are independent and the numbers of transformations on different layers are multiplied to obtain the order of the group. Finally, the set of all such transformations forms a group because, first, it is a subset of the finite symmetry group of a coordinate-axis-aligned cube centered at the origin in weight space, and second, because it is closed under composition (i.e., the product of any two transformations in the set is another transformation in the set). □

Thus, the set of all weight space transformations that leave the network input–output function unchanged forms an algebraic group G. We now analyze the structure of this group, beginning with a definition.

Definition 2. The group $\square_k$ is the set of transformations on the vector space $\mathbf{R}^k$ generated by changing the sign of any one coordinate component and by interchanging any two coordinate components.

The $\square$ notation is used because $\square_k$ is isomorphic to the symmetry group for a cube in k-dimensional space. An important fact about $\square_k$ is that it is the Weyl group of the n-dimensional classical Lie algebra B_k (Humphreys 1972; Helgason 1962; Weyl 1946). For the results presented here the most important property of the group $\square_k$ is the fact that it can be represented as the set of reflections generated by the roots of the specific Lie algebra B_k with which it is associated. From Weyl group theory (Humphreys 1972), the order of the group $\square_k$ is $k!\ 2^k$—which is exactly the size of the set of interchange and sign flip equioutput transformations for a hidden layer with k units. Thus, G might be constructible from Weyl groups. We now show that this suspicion is correct.

Theorem 3. *The group G is isomorphic to $\square_{M_2} \times \square_{M_3} \times \cdots \times \square_{M_{K-1}}$.*

Proof. Write the weight space W as

$$B_2 \times U_2 \times B_3 \times U_3 \times \cdots \times B_K \times U_K,$$

where B_l is the subspace of bias weights of layer l, and U_l is the subspace of nonbias weights of layer l. The group action can then be expressed for each hidden layer as the direct operation of the cube symmetry group $\square_{M_l}$ on each subspace B_l and as the indirect, but isomorphic, operation of nonbias weight interchanges and sign flips on the subspaces U_l and $U_{(l+1)}$. Only the hidden layers have symmetry groups associated with them, since the input and output layer units cannot be permuted.

Thus, each hidden layer contributes exactly one cube symmetry group to the overall group action. The group is isomorphic to the direct product of these groups because the actions of the individual groups operating on different layers commute. □

4 Search Sets

In this section we consider minimal sufficient search sets for multilayer perceptron neural networks. First, we provide some definitions that will be needed.

Definition 3. Two network weight vectors **u** and **v** in W are *equivalent* iff there exists $g \in G$ such that $g(\mathbf{u}) = \mathbf{v}$.

Definition 4. A *minimal sufficient search set* is a subset S of W such that each **w** in W is equivalent to exactly one element in S.

Definition 5. An *open minimal sufficient search set* is the interior of a minimal sufficient search set whose closure is equal to the closure of its interior.

Previous work by Hecht-Nielsen (1990) and Hecht-Nielsen and Evans (1991) demonstrated that there exist reasonably small, but nonminimal, sufficient search sets in the form of a cone bounded by hyperplanes. We now improve on this earlier work by showing that there exist open minimal sufficient search sets in the geometric forms of a wedge and a cone bounded by hyperplanes. Further, we supply formulas for these sets in terms of simple linear and quadratic inequalities, respectively.

Theorem 4. *The wedge interior described by the inequalities*

$$\begin{aligned} w_{li0} &> 0 \quad \textit{for} \quad 1 \le i \le M_l, \quad 2 \le l \le (K-1) \\ w_{li0} - w_{l(i+1)0} &> 0 \quad \textit{for} \quad 1 \le i \le (M_l - 1), \quad 2 \le l \le (K-1) \end{aligned} \tag{4.1}$$

is an open minimal sufficient search set for weight space W, as is the cone interior described by

$$\begin{aligned} w_{K10}\, w_{li0} &> 0 \quad \text{for} \quad 1 \le i \le M_l, \quad 2 \le l \le (K-1) \\ w_{K10}\, (w_{li0} - w_{l(i+1)0}) &> 0 \quad \text{for} \quad 1 \le i \le (M_l - 1), \quad 2 \le l \le (K-1) \end{aligned} \tag{4.2}$$

Proof. We construct the wedge by piecing together the Weyl chambers in the subspaces B_l of W. The cone is then constructed from the wedge. First, to simplify the notation, we define $U \equiv U_2 \times U_3 \times \cdots \times U_K$ so we can rewrite our decomposition of weight space as $W = B_2 \times B_3 \times \cdots \times B_K \times U$.

To begin our proof we observe that, since the Weyl group $\square_{M_l}$ acts directly on B_l, B_l can be identified with the root space of the classical Lie algebra B_{M_l} (Varadarajan 1984). This identification is unique because this particular Weyl group acts directly only on the root space of this one Lie algebra. An open minimal search set for the action of $\square_{M_l}$ on the root space of B_{M_l} is an open convex subset of the root space known as a *Weyl chamber* (Varadarajan 1984; Humphreys 1972; Helgason 1962). We will use D_l to denote the corresponding subset of B_l.

To proceed, we shall need the following technical Results concerning compositions of a group G_1 directly acting on space V_1 with open minimal search set S_1, and a group G_2 directly acting on space V_2 with open minimal search set S_2. In particular:

1. Let $G_1 \times G_2$ act on $V_1 \times V_2$ coordinatewise. Then $S_1 \times S_2$ is an open minimal search set of $V_1 \times V_2$ under $G_1 \times G_2$.

2. If $G_1 = G_2 = G$ and $g \in G$ acts on $(\mathbf{v}_1, \mathbf{v}_2)$ by $g(\mathbf{v}_1, \mathbf{v}_2) = (g\mathbf{v}_1, g\mathbf{v}_2)$, then $S_1 \times V_2$ is an open minimal search set for $V_1 \times V_2$ under G_1.

The proofs, which are elementary, are omitted.

By applying Result 1 successively to $B_2 \times B_3$, then to $(B_2 \times B_3) \times B_4$, and so on, we see that $D_2 \times D_3 \times \cdots \times D_{(K-1)}$ is an open minimal sufficient search set for $B_2 \times B_3 \times \cdots \times B_{(K-1)}$. Applying Result 2 to $(B_2 \times B_3 \times \cdots \times B_{(K-1)}) \times (B_K \times U)$ then shows that $D_2 \times D_3 \times \cdots \times D_{(K-1)} \times B_K \times U$ is a minimal sufficient search set for W.

Having characterized an open minimal sufficient search set for W, we now use the fact from Lie algebra theory that a Weyl chamber in the root space of B_k is determined by the inequalities $\alpha \cdot \mathbf{w} > 0$, where α is the Riesz representation vector for a positive root of the algebra (Varadarajan 1984; Humphreys 1972; Helgason 1962; Bachman and Narici 1966). For Lie algebra B_k there are k^2 positive roots of the form $\pm\hat{e}_i$ and $\pm(\hat{e}_i \pm \hat{e}_j)$, where the $\hat{e}_i$ are basis vectors in root space. We choose the roots $\hat{e}_i$ and $\hat{e}_i \pm \hat{e}_j$, with $i < j$, to be positive. Identifying these basis vectors with the hidden layer bias weight space positive coordinate axes gives us the following three sets of equations

$$\begin{aligned} w_{li0} &> 0 \quad \text{for} \quad 1 \le i \le M_l, \quad 2 \le l \le (K-1) \\ w_{li0} - w_{lj0} &> 0 \quad \text{for} \quad 1 \le i < j \le M_l, \quad 2 \le l \le (K-1) \\ w_{li0} + w_{lj0} &> 0 \quad \text{for} \quad 1 \le i < j \le M_l, \quad 2 \le l \le (K-1) \end{aligned}$$

However, the last set of inequalities is redundant, given the first set. Also, some of the inequalities in the second set are redundant. For example, if $w_{l30} > w_{l40}$ and $w_{l40} > w_{l50}$, then there is no need for the condition that $w_{l30} > w_{l50}$. Thus, we are left with inequalities 4.1 describing an open minimal sufficient search set in the form of a wedge. Note that this is a *wedge* because any positive real multiple of a member of the set is also a member of the set (as opposed to a *cone*, for which any real multiple must also be in the set).

The cone interior described by inequalities 4.2 is constructed by simply breaking the wedge across the hyperplane through the origin perpendicular to the bias weight axis of the first output layer unit, throwing away the portion of the wedge that intersects this hyperplane, and then rotating the bottom (negative) half-space half of the broken wedge by 180°. We can do this since the bias weights of the output units are unaffected by the group. □

Note that the wedge equations can be summarized by the simple statement that the weight vectors within (in the interior of) the fundamental wedge have all of their hidden layer bias weights positive and that the bias weight of each hidden unit is larger than the bias weight of the unit directly to its right. A similar statement holds for the cone. Also note that to turn these open minimal sufficient search sets into minimal sufficient search sets, we would have to add certain (but not all) points on their boundaries. For example, points with $w_{li0} = w_{lj0}$ for all units of each hidden layer would have to be added. Thus, for practical applications, we might simply want to use $\geq$ inequalities in 4.1 and 4.2.

Note that the images of a minimal sufficient search set S under different transformations in G are, except for certain points on their boundaries, disjoint. The entire weight space thus consists of the union of these sets

$$W = \bigcup_{g \in G} g[S]$$

As a result of this fact, and of the fact that the equioutput transformations themselves (namely, the elements of G) preserve not only the output of the network for all inputs but, thereby, the value of the network performance function as well, the network performance function is uniquely determined everywhere by its behavior on a minimal sufficient search set.

Unfortunately, the manner in which the behavior of a performance function in a wedge copy is determined from the behavior of that function within the fundamental wedge is not simple, since the hyperplanes that bound the wedge are not planes of symmetry for the transformations of G. That they are not is easy to see, since if $\mathbf{w}$ and $\mathbf{w}'$ are weight vectors that have all of their components equal except for one hidden unit bias weight differing in sign, or the bias weights of two adjacent hidden units interchanged (i.e., points placed symmetrically with respect to one of the bounding hyperplanes of the fundamental wedge), then, in general, there will be no transformation $g \in G$ such that $g(\mathbf{w})$ will equal $\mathbf{w}'$ (since the other weights involved in a sign flip or interchange transformation are not properly modified by this hyperplane reflection). Thus, in general, $F[g(\mathbf{w})]$ will not equal $F(\mathbf{w}')$. Understanding the relationship between the symmetries of G and the geometry of the fundamental wedge (or other minimal sufficient search sets) would seem to be an issue worthy of further investigation.

In this section we have examined one ramification of the geometric structure of weight space. Namely, the fact that searches for optimum weight vector values can be confined to an extremely small portion of the weight space. In the next section we consider another ramification of the group G.

5 Spherical Throngs of Optima

Another fact about the transformations in the group G is that they are isometries. Thus, for any $\mathbf{w} \in W$ and any $g \in G$,

$$|g(\mathbf{w})| = |\mathbf{w}|$$

That this is so, is easy to see because the only effect of applying any combination of sign flips and interchanges is to change the signs and permute the components of the weight vector, neither of which affect the Euclidean length of the vector.

Given that the elements of G are isometries, if $\mathbf{w}^*$ is a finite weight vector which optimizes the network performance function, then the points obtained by applying the elements of G to $\mathbf{w}^*$ all lie on the same sphere around the origin in weight space. In general, these points are all different and there are $\#G$ of them. We call such a set of points a *spherical throng of optima*.

Note that the copies of an optimal weight vector $\mathbf{w}^*$ in its spherical throng will, on average, have half of their weights modified from their corresponding values in $\mathbf{w}^*$ by a transformation in G. This is easy to see, since half of the permutation and sign flip compositions change more than half of the weights and half fewer than half. Thus, the copies of $\mathbf{w}^*$ in its throng will, in general, be scattered all over the sphere. Of course, in rare cases (such as where all of the weights have the same value), these copies are not scattered widely; but in general they are. It is easy to see that, given any member $\mathbf{w}$ of the throng, the nearest neighbor member on the sphere will typically be a vector that is the same as $\mathbf{w}$ except for a single sign flip or interchange transformation. Thus, nearest neighbors in the throng will tend to have most of their components the same.

The areal density of a spherical throng of optima depends upon the magnitude of $\mathbf{w}^*$, since the number of members of the throng is, in general, $\#G$, which depends only on the architecture of the network. If this magnitude ($|\mathbf{w}^*|$) is small, then the density will be high. Extensive experience with multilayer perceptrons at HNC, Inc. has shown that the lengths of apparently near-optimal weight vectors rarely, if ever, have lengths greater than $\sqrt{q}$, where again q is the number of weights in the network. In other words, the rms weight values in near-optimum weight vectors are typically considerably less than 1.0 in magnitude (exactly why this is so is unknown). Thus, it might be that these throngs are not only large, but often dense as well.

6 Implications

In this section we consider some implications of the results of Sections 2, 3, 4, and 5.

The existence of simple formulas for minimal sufficient search sets raises the question of whether such formulas will be of use in learning. They probably will not. In the case of gradient descent learning they would not be of use, since if we are going downhill, and happen to cross the boundary of the minimal sufficient search set, we should just continue the descent. Even if we wanted to move to the corresponding point of the performance surface within the fundamental wedge we could not do so, since (as pointed out in Section 4) we do not yet have a formula for finding this point.

For learning methods that employ stochastic jumping, rule-based weight modification, or another nongradient descent method, it might seem to be of use to constrain the hidden layer bias weights so as to force the network weight vector to remain within the fundamental wedge (or some other minimal sufficient search set). However, this is not really true, as the following example shows.

Imagine a simple performance surface with one and only one finite minimum (located near, but not at, the origin) within the fundamental wedge. The goal is to find a weight vector within ϵ distance of this minimum. Suppose that a simple unconstrained discrete-time gaussian random weight space search were being used to find this minimum. Then there would appear to be a search speed-up of $\#G$ to be gained by constraining the search to an equivalent search process within a minimal sufficient search set. However, this is an illusion, because the unconstrained search process is not trying to find a single minimum (as the constrained process is). It need only find one of $\#G$ equivalent copies of the minimum. Therefore, both searches will have the same expected number of steps.

Thus, we conclude that knowing the geometry of a minimal sufficient search set has no obvious benefit for learning.

With respect to spherical throngs of optima, we speculate that gradient descent learning may be aided by the fact that most learning procedures follow the tradition of starting the weight vector components at random values chosen over a small interval centered at zero.

Starting near the origin has more than one benefit. First, starting near the origin causes the "approximating surface" of the network to start out nearly "flat"—with its initial output value near zero everywhere. As the training process proceeds this initially flat approximating surface begins to "crinkle up" as it tries to fit the training data. Thus, starting the weight values near zero provides a parsimonious surface of initially nearly zero curvature. Another reason why this tradition is so apt is that the usual activation functions *tanh* and $(1+e^{-u})^{-1}$ have all of their significant behavior near zero. Thus, one would naturally expect that, if the inputs to the network tend to be small in magnitude, large weight values would be needed only rarely. As mentioned above, anecdotal evidence suggests that this is what occurs in many practical situations.

Geometrically, when gradient descent training begins with an initial weight vector near the origin, we conjecture that the learning process consists of an initial, generally outward, movement from the origin to a radius at which a spherical throng of optima is encountered, followed by a "homing in" process that guides the weight vector toward an optimum. If this conjecture is correct, and if, as we suspect, many practical problems have a performance surface with a spherical throng of optima located at a relatively small radius, then this dense shell of optima may be a relatively "easy" target to hit. In other words, in contrast to the typical optimization situation (e.g., in linear programming, combinatorial optimization, or unconstrained optimization), where we are searching a high-dimensional space for a single optimum (or one of a small number of optima), here we are searching a high-dimensional space for any one of a vast multitude of optima. This may partially explain why the training of multilayer perceptron networks with thousands or even millions of adaptive weights is often practically feasible (a fact that we now take for granted, but which, a priori, is rather surprising).

Acknowledgments

We thank the referees for several helpful comments, including the observation that discontinuous equioutput transformations can exist (this fact was also pointed out to us by Arcady Nemirovskiĭ). Thanks also to Luis Almeida, Charles Fefferman, Lee Giles, David Hestenes, Margarita Kuzmina, Irina Surina, and Robert Williamson for valuable discussions.

References

Bachman, G., and Narici, L. 1966. *Functional Analysis.* Academic Press, New York.

Becker, S., and Hinton, G. E. 1992. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature (London)* **355**, 161–163.

Bishop, C. M. 1991. Improving the generalization properties of radial basis function neural networks. *Neural Comp.* **3**, 579–588.

Bishop, C. M. 1990. Curvature-driven smoothing in backpropagation neural networks. *Proc. of the International Neural Network Conf., Paris.* **2**, 749–752. Kluwer, Dordrecht.

Broomhead, D. S., and Lowe, D. 1988. Multivariable function interpolation and adaptive networks. *Complex Syst.* **2**, 321–355.

Carpenter, G. A., Grossberg, S., and Reynolds, J. H. 1991. ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks* **4**, 565–588.

Chen, A. M., and Hecht-Nielsen, R. 1991. On the geometry of feedforward neural network weight spaces. *Proc. Second IEE International Conference on Neural Networks,* 1–4. IEE Press, London.

Hartmann, E. J., Keeler, J. D., and Kowalski, J. M. 1990. Layered neural networks with gaussian hidden units as universal approximations. *Neural Comp.* **2**, 210–215.

Hecht-Nielsen, R. 1992. Theory of the backpropagation neural network. In *Neural Networks for Human and Machine Perception,* Volume 2, H. Wechsler, ed., pp. 65–93. Academic Press, Boston, MA.

Hecht-Nielsen, R. 1991. *Neurocomputing.* Addison-Wesley, Reading, MA.

Hecht-Nielsen, R. 1990. On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers,* R. Eckmiller, ed. Elsevier-North Holland, Amsterdam.

Hecht-Nielsen, R., and Evans, K. M. 1991. A method for error surface analysis. In *Theoretical Aspects of Neurocomputing,* M. Novák and E. Pelikán, eds., pp. 13–18. World Scientific, Singapore.

Helgason, S. 1962. *Differential Geometry and Symmetric Spaces.* Academic Press, New York.

Humphreys, J. E. 1972. *Introduction to Lie Algebras and Representation Theory.* Springer-Verlag, New York.

Linsker, R. 1988. Self-organization in a perceptual network. *IEEE Computer Mag.* **21**, 105–117.

Moody, J., and Darken, C. J. 1989. Fast learning in networks of locally-tuned processing units. *Neural Comp.* **1**, 281–294.

Poggio, T., and Girosi, F. 1990. Regularization algorithms for learning that are equivalent to multilayer networks. *Science* **247**, 978-982.

Reilly, D. L., Cooper, L. N., and Elbaum, C. 1982. A neural model for category learning. *Biol. Cyber.* **45**, 35–41.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. 1986. Learning internal representations by error propagation. In *Parallel Distributed Processing,* D. E. Rumelhart and J. L. McClelland, eds. MIT Press, Cambridge, MA.

Sussmann, H. J. 1992. Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural Networks* **5**, 589–593.

Varadarajan, V. S. 1984. *Lie Groups, Lie Algebras, and Their Representations.* Springer-Verlag, New York.

Weyl, H. 1946. *The Classical Groups.* Princeton University Press, Princeton.

Received 27 March 1992; accepted 15 March 1993.