Neural Networks, Vol. 5, pp. 755–764, 1992
Printed in the USA.

0893-6080/92 $5.00 + .00

ORIGINAL CONTRIBUTION

Optimization of the Hidden Unit Function in Feedforward Neural Networks

OSAMU FUJITA

NTT LSI Laboratories

(*Received* 6 *February* 1991; *revised and accepted* 26 *December* 1991)

Abstract—*A novel objective function is proposed for optimizing the hidden unit function in feedforward neural networks. This objective function represents the performance of the hidden unit at minimizing the least squared output errors of the linear output unit. This is derived from the decrease in the output errors due to the addition of the hidden units. The optimized output state vectors of the hidden units span a proper state space, which includes the desired output vectors for the network. The optimization (maximization of the objective function) is equal to minimizing the angle between the desired output vector and the projection of the hidden unit's output state vector onto the orthogonal complement of the subspace spanned by the other state vectors. The approximate solution can be obtained using the gradient ascent algorithm. This optimization method is useful in constructing fully connected feedforward neural networks and for minimizing the size of layered networks.*

Keywords—Hidden unit function, Least squared errors, Optimum state space, Projection matrix, Gradient ascent algorithm, Feedforward network synthesis.

1. INTRODUCTION

The hidden units in feedforward neural networks perform significant nonlinear data transformation for output units in order to produce arbitrary output functions. The number of hidden units is closely related to the network capacity (Akaho & Amari, 1990; Baum, 1988). When the number of hidden units is too small to produce a desired output function, more hidden units are needed. On the other hand, too many hidden units actually lower the generalization ability of the network, according to Akaike's Information Criterion (AIC) (Akaike, 1974; Kurita, 1990). Excess hidden units must be removed and the remains must be optimized. It is necessary to evaluate the performance of each hidden unit, whether adding or removing hidden units. The problem is what function is required for each hidden unit and how should it be evaluated to improve the performance.

In network synthesis via Back-Propagation Learning (BPL) (Rumelhart & McClelland, 1986), the network structure is usually fixed, even though the necessary number of hidden units is not clear *a priori*. Convergence to a satisfiable solution is not always guaranteed. To improve the convergency and to speed it up, techniques have been investigated for adjusting the proper number of hidden units by adding or removing hidden units during the learning process (Ash, 1989; Hagiwara, 1990; Hirose, Yamashita, & Hijiya, 1989). Such a practical technique has been experimentally shown to be effective for convergence, but the reasons for this success have not been clarified.

A network of hidden units can be constructed using many other strategies that do not use BPL. Some examples include the Cascade-Correlation Learning architecture (CCL) (Fahlman & Lebiere; 1989), the pocket algorithm (Gallant, 1990), the tiling algorithm (Mezard & Nadal, 1989), the saturated projection algorithm (Kawahara & Irino, 1988), recruitment learning techniques (Deffuant, 1990; Diederich, 1988), geometrical methods (Hopcroft & Mattson, 1965; Ramacher & Wesseling, 1989; Rujan & Marchand, 1989), and the orthogonal complement method (Fujita, 1990). These methods add hidden units according to demand. CCL makes additional hidden units so as to cancel residual output errors of the network. Some of the others are able to design the hidden units to have a specific function using *a priori* knowledge of input-output relationships. If there is no conflicting relationship in

Acknowledgement: The author wishes to thank Professor Shunichi Amari, Dr. Iwao Toda, Dr. Hideki Kawahara, Dr. Atsushi Iwata, and Dr. Yoshihito Amemiya for their comments and suggestions.

Requests for reprints should be sent to O. Fujita, NTT LSI Laboratories, 3-1, Morinosato Wakamiya, Atsugi-shi, Kanagawa Pref., 243-01, Japan.

sample data sets and no limit to the number of units, such a constructive algorithm can always produce networks that perform arbitrary desired functions. An important problem is how to optimize the network structure so that it does not become too large.

This paper proposes a novel objective function for optimizing the hidden units, in order to give a minimum network the best performance, and shows how the function has been applied to network synthesis. The objective function can be derived from the decrease in the least squared output-errors of a network due to the addition of one hidden unit. Maximization of the objective function for each hidden unit provides a "one shot" learning method and it is very useful for constructing a minimum network. The theory is based on the least squares approximation using the pseudoinverse or projection matrices, as already used for linear classifiers (Duda & Hart, 1973; Ho & Kashyap, 1965; Webb & Lowe, 1990). This optimization deals with given data sets only, but it is applicable to random sampling data sets, so it is usable for adaptive learning, in principle, according to a generalized stochastic learning scheme (Amari, 1967). The optimization does not directly aim at generalization ability for dealing with unspecified data sets properly, but it is closely connected with the principle of parsimony appearing in AIC so it may have a good effect on generalization.

This paper is organized as follows: Section 2 gives a basic definition of the problem, and Section 3 gives a short summary of the least squares approximation. The objective function is introduced in Section 4 and discussed in order to obtain an optimum solution. The network synthesis and experimental results are described in Section 5. Optimization for nonlinear output units is discussed in Section 6. The advantages and disadvantages of this method are discussed in Section 7.

2. STATEMENT OF THE PROBLEM

Let us consider a neural network with n variable inputs and one output. Suppose that K data sets for the inputs and the desired output are given. The problem is how to determine the internal structure of the network in order to satisfy the input-output relationships.

The input data sets and the desired output data sets can be represented by a $K \times n$ matrix $\mathbf{X}$ and a K-dimensional vector $\mathbf{z}$, respectively. The neural network has internal states represented by a $K \times m$ matrix $\mathbf{H}$, whose elements represent outputs of m hidden units, produced by a nonlinear transformation of $\mathbf{X}$ as follows:

$$\mathbf{H} = f(\mathbf{X}). \tag{1}$$

The nonlinear function f is usually defined by a sigmoid function such as $H_{ij} = \tanh(\mathbf{v}_i \mathbf{w}_j)$ where $\mathbf{v}_i$ is the i-th row vector of $\mathbf{X}$ and $\mathbf{w}_j$ is an n-dimensional weight (coefficient) vector for a modifiable parameter of f. Another type of function such as $H_{ij} = \exp(-\|\mathbf{v}_i - \mathbf{w}_j\|^2)$ and $H_{ij} = \sin(\mathbf{v}_i \mathbf{w}_j)$ is also available.

Let $\mathbf{y}$ be an actual output vector produced by a linear combination of the column vectors of $\mathbf{X}$ and $\mathbf{H}$. Thus, $\mathbf{y}$ satisfies

$$\mathbf{y} \in L[\mathbf{XH}], \tag{2}$$

where $L[\mathbf{XH}]$ denotes the column space of the augmented matrix $[\mathbf{XH}]$. Moreover, $\mathbf{y}$ should also satisfy the condition that $\mathbf{y}$ is approximately equal to $\mathbf{z}$,

$$\mathbf{y} \approx \mathbf{z}. \tag{3}$$

In practice, for example, the Condition (3) is defined by

$$\text{C1: } \|\mathbf{z} - \mathbf{y}\|_2 < \epsilon,$$

$$\text{C2: } \|\mathbf{z} - \mathbf{y}\|_\infty < \epsilon, \quad \text{or}$$

$$\text{C3: } \min\{y_i z_i\} > \alpha,$$

where ϵ and α are positive constants. C1 and C2 are defined by the L_2 norm (the Euclidean norm) and L_∞ norm (the maximum error), respectively. C3 is useful for a binary output condition such that $z_i \in \{1, -1\}$, because a wide range of y_i satisfies C3 and such y_i can easily be transformed into z_i with the sign function. Although C1 and C2 can be a sufficient condition for C3, C1 is a severer criterion than C2, and C2 is severer than C3.

The problem thus becomes how to determine $\mathbf{H}$ that satisfies conditions (1), (2), and (3). However, there are many solutions, which include a trivial one of $\mathbf{H}$ consisting of K columns. To obtain a useful solution, let us consider the following objective.

OBJECTIVE. *Minimization of rank* $[\mathbf{XH}]$.

Minimization of *rank* $\mathbf{H}$ means minimization of the number of units and modifiable parameters. This objective is so important that it profoundly influences the generalization ability in connection with the principle of parsimony, which appears in the best statistical model choice using AIC. However, it is a tough problem which depends on the computational complexity of tasks.

3. LEAST SQUARES APPROXIMATION

When there are no internal states, $\mathbf{y}$ has to be obtained from the linear combination of column vectors $\mathbf{x}$ of $\mathbf{X}$ as follows:

$$\mathbf{X}\mathbf{w}_o = \mathbf{y}, \tag{4}$$

where $\mathbf{w}_o$ is a weight vector whose components are coefficients of $\mathbf{x}$. Using the Euclidean norm C1 for Condition (3), if

$$\|\mathbf{z} - \mathbf{y}\|^2 = \sum_{i=1}^{k} (z_i - y_i)^2, \tag{5}$$

is minimum, then $\mathbf{y}$ is optimum and given by

$$\mathbf{y} = \mathbf{P}\mathbf{z}, \tag{6}$$

where $\mathbf{P}$ is the projection matrix onto the column space of $\mathbf{X}$. Hence, the minimum norm is expressed by $\|\mathbf{z} - \mathbf{P}\mathbf{z}\| = \|\mathbf{P}_c\mathbf{z}\|$ where $\mathbf{P}_c = \mathbf{I} - \mathbf{P}$ and $\mathbf{I}$ is the identity matrix. $\mathbf{P}_c$ is the projection matrix onto the orthogonal complement of the column space of $\mathbf{X}$, and it has two basic properties, $\mathbf{P}_c^2 = \mathbf{P}_c$ and $\mathbf{P}_c^T = \mathbf{P}_c$, in common with $\mathbf{P}$. If $\mathbf{X}$ consists of linearly independent columns only, then $\mathbf{X}^T\mathbf{X}$ is nonsingular and $\mathbf{P}$ is expressed by

$$\mathbf{P} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T. \tag{7}$$

The optimum weight vector is given by $\mathbf{w}_o = \mathbf{X}^\dagger\mathbf{z}$, where $\mathbf{X}^\dagger = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ which is called the pseudo-inverse of $\mathbf{X}$.

The squared minimum norm, which is expressed as $\mathbf{z}^T\mathbf{P}_c\mathbf{z}$, is a basic function for evaluating the optimality of the internal representation of a multilayer classifier in which output transfer functions are linear (Webb & Lowe, 1990).

4. OPTIMIZATION OF AN ADDITIONAL HIDDEN UNIT

4.1. The Objective Function

Let us consider the case where $\mathbf{H}$ consists of only one column vector $\mathbf{h}$ (i.e., only one hidden unit is added to the network). The problem is to find the vector $\mathbf{h}$ that minimizes the norm $\|\mathbf{z} - \mathbf{y}\|$ when $\mathbf{y}$ belongs to the column space of $[\mathbf{X}\mathbf{h}]$. For that purpose, the following theorem gives an objective function for the optimization of $\mathbf{h}$.

THEOREM. *When the space to which* $\mathbf{y}$ *belongs is expanded from* $L[\mathbf{X}]$ *to* $L[\mathbf{X}\mathbf{h}]$, *the minimum value of* $\|\mathbf{z} - \mathbf{y}\|^2$ *decreases by* Δ, *which is given by*

$$\Delta = \frac{(\mathbf{z}^T\mathbf{P}_c\mathbf{h})^2}{(\mathbf{h}^T\mathbf{P}_c\mathbf{h})}. \tag{8}$$

Proof. Let $\mathbf{P}'$ be the projection matrix onto the column space of $[\mathbf{X}\mathbf{h}]$, i.e.,

$$\begin{aligned}\mathbf{P}' &= [\mathbf{X}\mathbf{h}]([\mathbf{X}\mathbf{h}]^T[\mathbf{X}\mathbf{h}])^{-1}[\mathbf{X}\mathbf{h}]^T \\ &= \mathbf{P} + \mathbf{P}\mathbf{h}C\mathbf{h}^T\mathbf{P} - \mathbf{P}\mathbf{h}C\mathbf{h}^T\mathbf{P} + \mathbf{h}C\mathbf{h}^T \\ &= \mathbf{P} + \mathbf{P}_c\mathbf{h}C\mathbf{h}^T\mathbf{P}_c, \end{aligned} \tag{9}$$

where $C = 1/(\mathbf{h}^T\mathbf{P}_c\mathbf{h})$ (see Appendix). Thus, $\mathbf{P}_c' = \mathbf{P}_c(\mathbf{I} - \mathbf{h}C\mathbf{h}^T)\mathbf{P}_c$. Therefore, Δ is given by

$$\begin{aligned}\Delta &= \|\mathbf{z} - \mathbf{P}\mathbf{z}\|^2 - \|\mathbf{z} - \mathbf{P}'\mathbf{z}\|^2 \\ &= \mathbf{z}^T(\mathbf{I} - \mathbf{P})\mathbf{z} - \mathbf{z}^T(\mathbf{I} - \mathbf{P}')\mathbf{z} \\ &= \mathbf{z}^T\mathbf{P}_c\mathbf{h}C\mathbf{h}^T\mathbf{P}_c\mathbf{z} \\ &= \frac{(\mathbf{z}^T\mathbf{P}_c\mathbf{h})^2}{(\mathbf{h}^T\mathbf{P}_c\mathbf{h})}. \end{aligned}$$

Q.E.D.

The problem is thus to find $\mathbf{h}$ that maximizes Δ. The Δ represents the performance or effectiveness of the hidden unit at reducing the output error and producing $\mathbf{y}$ such that $\mathbf{y} \approx \mathbf{z}$.

4.2. Optimization

The objective function Δ can be expressed in the form of Rayleigh's quotient as follows:

$$\Delta = \frac{(\mathbf{h}^T\mathbf{P}_c\mathbf{z}\mathbf{z}^T\mathbf{P}_c\mathbf{h})}{(\mathbf{h}^T\mathbf{P}_c\mathbf{P}_c\mathbf{h})}. \tag{10}$$

The maximum value of Δ is equal to the maximum eigenvalue of the matrix $\mathbf{P}_c\mathbf{z}\mathbf{z}^T\mathbf{P}_c$, if $\mathbf{P}_c\mathbf{h}$ is equal to an eigenvector such as $\mathbf{P}_c\mathbf{z}$. In a neural network, however, $\mathbf{P}_c\mathbf{h}$ cannot always be such an eigenvector in the orthogonal complement of $L[\mathbf{X}]$, because $\mathbf{h}$ is limited to the range of $f(\mathbf{X})$. Even if $\mathbf{P}_c\mathbf{h}$ is determined exactly, it is difficult to determine $\mathbf{h}$ from $\mathbf{P}_c\mathbf{h}$, because $\mathbf{P}_c$ is not invertible.

Considering the nonlinearity of eqn (1), in practice, only the approximation can be used to obtain a nearly optimal solution. For example, let us consider a nonlinear function expressed by

$$\mathbf{h} = \mathbf{f}(\mathbf{X}\mathbf{w}), \tag{11}$$

where $\mathbf{f}(\mathbf{a})$ is a vector nonlinearly transformed from $\mathbf{a}$ by a nonlinear function f with respect to each component (i.e., $\mathbf{f}(\mathbf{a}) = [f(a_1)\cdots f(a_k)]^T$), and $\mathbf{w}$ is the weight vector for the inputs of the additional hidden unit.

The objective function Δ can be maximized by using the gradient ascent algorithm in which w_i is successively changed by $\delta w_i \propto \partial\Delta/\partial w_i$ so as to increase Δ; however, the solution is sometimes trapped into a local maximum. The change in the weight vector, $\delta\mathbf{w}$, is given by

$$\delta\mathbf{w}^T = \eta\gamma(\mathbf{z}^T - \gamma\mathbf{h}^T)\mathbf{P}_c\delta\mathbf{H}, \tag{12}$$

where η is a positive constant, $\gamma = (\mathbf{z}^T\mathbf{P}_c\mathbf{h})/(\mathbf{h}^T\mathbf{P}_c\mathbf{h})$ and $\delta\mathbf{H}$ represents the derivatives of $\mathbf{h}$ with respect to w_i ($i = 1, \ldots n$) and is defined as follows:

$$\begin{aligned}\delta\mathbf{H} &= \left[\frac{\partial}{\partial w_1}\mathbf{h}\cdots\frac{\partial}{\partial w_n}\mathbf{h}\right] \\ &= [\mathbf{F}'(\mathbf{X}\mathbf{w})\mathbf{x}_1\cdots\mathbf{F}'(\mathbf{X}\mathbf{w})\mathbf{x}_n] \\ &= \mathbf{F}'(\mathbf{X}\mathbf{w})\mathbf{X}, \end{aligned} \tag{13}$$

where $\mathbf{F}'(\mathbf{a})$ is the diagonal matrix such that the i-th diagonal element is equal to the derivative $f'(a_i)$.

The objective function Δ can be rewritten in another form:

$$\Delta = \|\mathbf{z}\|^2\cos^2\theta, \tag{14}$$

where $\cos\theta = (\mathbf{z}^T\mathbf{P}_c\mathbf{h})/\|\mathbf{z}\|\|\mathbf{P}_c\mathbf{h}\|$. Thus, this optimization indicates a decrease in the angle θ between $\mathbf{z}$ and $\mathbf{P}_c\mathbf{h}$ which is in the orthogonal complement of the column space of $\mathbf{X}$ (or in the left null space of $\mathbf{X}$), as shown in Figure 1.

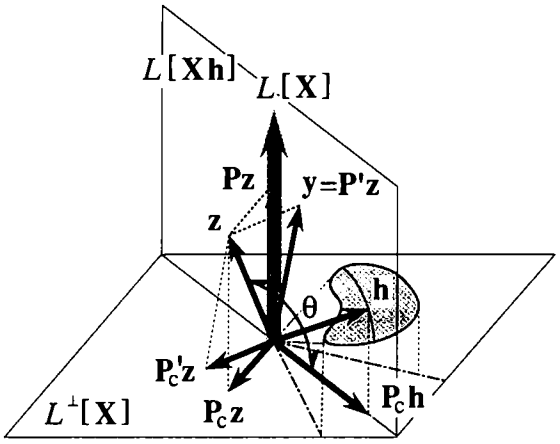


FIGURE 1. Schematic diagram of the *K*-dimensional state space. For convenience, the *n*-dimensional subspace *L*[X] and its orthogonal complement $L^{\perp}$[X] are represented by the one-dimensional vertical vector and the horizontal plane, respectively. The gray regions around h and *Pch* represent the range of *f*(*X*) and its projection onto $L^{\perp}$[X], respectively. The optimal h is a vector that minimizes the angle θ between z and Pch.

For multiple output networks, the objective function should be the summation of Δ's for all desired output vectors, which can be expressed by (10) where $\mathbf{z}$ is replaced by the matrix $\mathbf{Z}$ consisting of all $\mathbf{z}$ as column vectors.

5. FEEDFORWARD NETWORK SYNTHESIS

5.1. Fully Connected Feedforward Network

The network structure of the fully connected feedforward network is shown in Figure 2, as an example. The hidden units are numbered from 1 to m. The i-th hidden unit can receive the outputs of any lower-numbered hidden units 1 to i-1. The state vectors satisfy the following conditions:

$$\mathbf{y} \approx \mathbf{z} \quad \text{and} \quad \mathbf{y} \in L[\mathbf{XH}],$$

for the output unit, and

$$\mathbf{h}_i = \mathbf{f}(\mathbf{y}_i) \quad \text{and} \quad \mathbf{y}_i \in L[\mathbf{XH}_i] \quad (i = 1, \ldots, m),$$

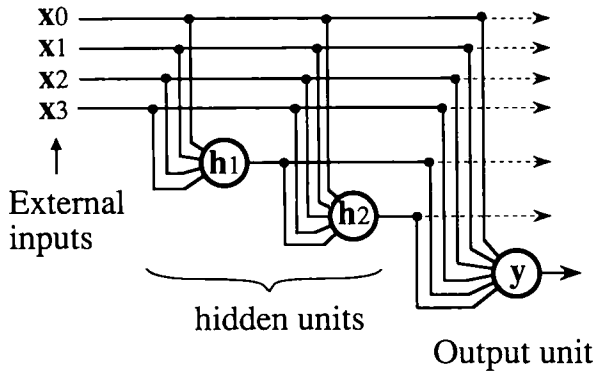


FIGURE 2. Fully connected feedforward network (for *n* = 4 and *m* = 2).

for the hidden units, where

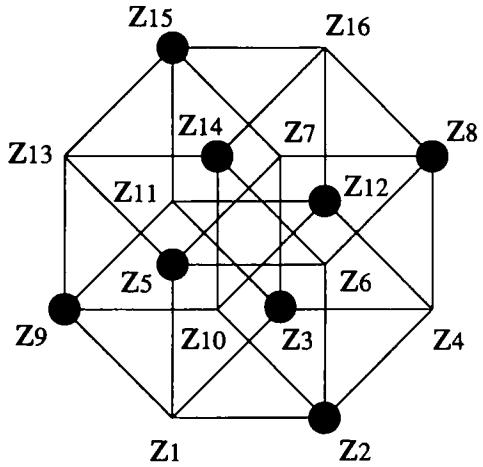
$$\mathbf{H}_i = [\mathbf{h}_1 \cdots \mathbf{h}_{i-1}] \quad \text{(Type 1)},$$

except for $\mathbf{H}_1 = [\,]$ (no elements). Using all state vectors to span the maximum state space $L[\mathbf{XH}]$, the network can produce $\mathbf{y} \approx \mathbf{z}$ with a minimum number of hidden units. Thus, this type of the network has the greatest capacity for producing desired output functions. For example, the parity function of n variables can be implemented by using $[\log_2 n]$ hidden units for a threshold logic network (TLN) (Kautz, 1961).

The state vector $\mathbf{h}_i$ can be determined in the order of the unit's number i. The hidden units are added one by one according to demand, until $\|\mathbf{z} - \mathbf{y}\|$ becomes zero. In practice, the criterion $\|\mathbf{z} - \mathbf{y}\| < 1$ is a sufficient condition for the threshold output unit to produce binary outputs $\{1, -1\}$. The state matrix $[\mathbf{XH}_i]$ is augmented with adding $\mathbf{h}_i$ step by step, so the projection matrix $\mathbf{P}$ onto $L[\mathbf{XH}_i]$ changes according to a relation like eqn (7). However, $\mathbf{P}$ can easily be calculated from the recurrence relation (9) without matrix inversion. The total computation time for designing this type of network is less than that for layered networks discussed in the following section.

The network synthesis strategy is much the same with CCL (Fahlman & Lebiere, 1990), but there is a difference in the objective function to be maximized. The CCL is based on maximization of a quantity S, a kind of the covariance between residual output errors and hidden unit outputs, instead of Δ. It can be said that the hidden units are formed to compensate for residual output errors. If output units have a linear function and $\mathbf{y} = \mathbf{Pz}$, then the residual output errors can be expressed by $\mathbf{P}_c\mathbf{z}$, and so S is nearly equal to $|\mathbf{z}^T\mathbf{P}_c\mathbf{h}|$ in a special case. In this case, the major difference between Δ and S is the factor $C = 1/(\mathbf{h}^T\mathbf{P}_c\mathbf{h})$. Maximizing S tends to make $\mathbf{h}$ approach $\mathbf{P}_c\mathbf{z}$, which probably gives a different solution. Which is better depends on the application.

Experiment 1. Computational experiments of network synthesis have had good results for the network that performs four-variable Boolean functions (i.e., the mapping of $\{1, 0\}^4 \rightarrow \{1, 0\}$). Those functions, which total up to 2^{16}, can be represented by a pattern of binary numbers which are assigned to the vertices of a four-dimensional hypercube in the row space of $\mathbf{X}$, and classified into 238 symmetry classes (Golomb, 1959; Slepian, 1954). For example, the 4-bit parity function is shown in Figure 3. Functions belonging to the same class are equal in complexity of the pattern and network implementation. For a representative function in each symmetry class, the necessary number (m) of hidden units designed by the proposed method is evaluated in comparison with the ideally minimum number (m_i) for TLN.



$$\mathbf{X}_0^T = [\,-1\ -1\ -1\ -1\ \ -1\ -1\ -1\ -1\ \ -1\ -1\ -1\ -1\ \ -1\ -1\ -1\ -1\,]$$
$$\mathbf{X}_1^T = [\,-1\ -1\ -1\ -1\ \ -1\ -1\ -1\ -1\ \ 1\ 1\ 1\ 1\ \ 1\ 1\ 1\ 1\,]$$
$$\mathbf{X}_2^T = [\,-1\ -1\ -1\ -1\ \ 1\ 1\ 1\ 1\ \ -1\ -1\ -1\ -1\ \ 1\ 1\ 1\ 1\,]$$
$$\mathbf{X}_3^T = [\,-1\ -1\ 1\ 1\ \ -1\ -1\ 1\ 1\ \ -1\ -1\ 1\ 1\ \ -1\ -1\ 1\ 1\,]$$
$$\mathbf{X}_4^T = [\,-1\ 1\ -1\ 1\ \ -1\ 1\ -1\ 1\ \ -1\ 1\ -1\ 1\ \ -1\ 1\ -1\ 1\,]$$

$$\mathbf{Z}^T = [\,-1\ 1\ 1\ -1\ \ 1\ -1\ -1\ 1\ \ 1\ -1\ -1\ 1\ \ -1\ 1\ 1\ -1\,]$$

$Z_1\ Z_2\ Z_3\ Z_4\ \cdots \qquad \cdots\ Z_{16}$

FIGURE 3. The 4-bit parity function represented in the 4-dimensional hypercube diagram. The input state vectors and the desired output vector are also shown as $x_{0}, \ldots x_{4}$, and z. The *i*-th vertex corresponds with the *i*-th components of the state vectors. The closed circle on the *i*-th vertex indicates $z_i = 1$. Parity function implementation is easy for the fully connected feed-forward network, but difficult for the layered network.

The input and desired output data were given $\{1, -1\}$ instead of $\{1, 0\}$. The inputs consisted of four variable inputs and one constant input, so $\mathbf{X}$ was a $2^4 \times 5$ matrix. Hence, $\mathbf{z}$ and $\mathbf{y}$ were 2^4-dimensional vectors. The nonlinear function f was defined as $h_i = \tanh(y_i)$. The design criterion was C1: $\|\mathbf{z} - \mathbf{y}\| < 0.1$.

The $\mathbf{h}$ vectors were modified with $\eta = 7.0$ until the decreasing rate of $\mathbf{z}^T\mathbf{P}_c\mathbf{z}$ became lower than a certain rate. In the maximization of Δ, there are many local maxima, and so several trials starting from various initial conditions of $\mathbf{w}$ were needed to obtain the best one for $\mathbf{h}_i$. The best was chosen from among five candidates and was easily obtained when the initial $\mathbf{w}$ was near in the direction of $[\mathbf{X}\mathbf{H}_i]^\dagger\mathbf{z}$.

Table 1 shows the distribution of the 238 functions with respect to both the ideally minimum necessary number of hidden units and the actual number designed by the proposed method. For example, there are 179 (=2 + 155 + 22) functions that can be ideally implemented by the minimum network having one hidden unit. For the 155 functions, one more hidden unit is needed for the network designed by the proposed method. For every function, the number of hidden units can be designed to be closely minimum and not more than three.

The results for CCL in similar experiments are also shown in Table 1. Both methods achieve almost comparable results. They differ slightly in the residual error at each stage in the addition of hidden units as shown in Table 2. The proposed method can give better $\mathbf{h}_1$ than CCL, but an optimum $\mathbf{h}_1$ does not necessarily produce optimum $\mathbf{h}_2$. An advantage of CCL is that the convergence to an optimum $\mathbf{h}$ is less sensitive to initial conditions.

The average number of iterations required for convergence was 27 for the proposed method and 108 for CCL, but CCL is simpler and faster to compute than the proposed method. The total computation time for all the 238 functions in the case that each $\mathbf{h}$ was chosen from among five candidates was 150s for the proposed method and 84s for CCL on a mainframe computer. However, it should be noted that the results varied greatly with initial conditions, constant values, and criteria for iteration control.

TABLE 1
Distribution of the Representative Boolean Functions with Respect to the Minimum Number of Hidden Units in Experiment 1

		Proposed Method					CCL				
mi	*Nf*	*m* = 0	=1	=2	=3	=4	*mc* = 0	=1	=2	=3	=4
=0	15	2	13	0	0	0	2	13	0	0	0
=1	179	0	2	155	22	0	0	2	159	16	2
=2	44	0	0	4	40	0	0	0	5	39	0
Total	238	2	15	159	62	0	2	15	164	55	2

Nf: number of representative Boolean functions.
mi: ideally minimum number of hidden units for TLN.
m, *mc*: number of hidden units designed by the proposed method or CCL.

TABLE 2
Residual Error in Experiment 1

		Proposed Method		CCL	
mi	*m*, *mc* = 0	*m* = 1	*m* = 2	*mc* = 1	*mc* = 2
=2	10.8 ± 2.6	3.5 ± 1.2	—	3.7 ± 1.1	—
=3	13.3 ± 1.6	5.7 ± 1.3	2.3 ± 0.8	5.9 ± 1.2	2.5 ± 1.0

Residual errors are expressed by (average) ± (standard deviation).

5.2. Layered Feedforward Network

The structure of the layered feed-forward network is shown in Figure 4. If there is only one hidden layer, the state vectors satisfy the following conditions:

$$\mathbf{y} \approx \mathbf{z} \quad \text{and} \quad \mathbf{y} \in L[\mathbf{H}],$$

and

$$\mathbf{h}_i = \mathbf{f}(\mathbf{y}_i) \quad \text{and} \quad \mathbf{y}_i \in L[\mathbf{X}], \quad (i = 1, \ldots, m).$$

This is a special case of the fully connected feedforward networks where some connections are cut (i.e., the weights are fixed at zero). Thus, the network can be constructed by adding optimized $\mathbf{h}_i$ one by one as mentioned above.

Beside this, there is another way to optimize all $\mathbf{h}_i$ simultaneously for a fixed-size network. Initially, m hidden units are provided for the network, and then each hidden unit is modified in turn to increase Δ, as if it were an additional unit. The objective function Δ_i for each $\mathbf{h}_i$ varies with

$$\mathbf{P}_i = \mathbf{H}_i(\mathbf{H}_i^T\mathbf{H}_i)^{-1}\mathbf{H}_i^T,$$

where

$$\mathbf{H}_i = [\mathbf{h}_1 \cdots \mathbf{h}_{i-1}\ \mathbf{h}_{i+1} \cdots \mathbf{h}_m] \quad \text{(Type 2)},$$

which is different from $\mathbf{H}_i$ of Type 1 shown in Section 5.1. A drawback is that it takes a lot of time to compute $\mathbf{P}_i$, because $\mathbf{P}_i$ changes with $\mathbf{h}_j$ ($i \neq j$) every step. Time is mainly wasted on calculating the inverse matrix $(\mathbf{H}_i^T\mathbf{H}_i)^{-1}$. To avoid tedious calculation, larger networks will have to be approximated: for example, (a) omitting off-diagonal elements of $\mathbf{H}_i^T\mathbf{H}_i$ as $\mathbf{h}_j^T\mathbf{h}_k = 0$ ($j \neq k$) when the $\mathbf{h}$ vectors are almost orthogonal; (b) dividing $\mathbf{H}_i$ into small submatrices such as $\mathbf{H}_{ijk} = [\mathbf{h}_j\mathbf{h}_k]$ ($i \neq j \neq k \neq i$) and maximizing the sum of Δ for each submatrix.

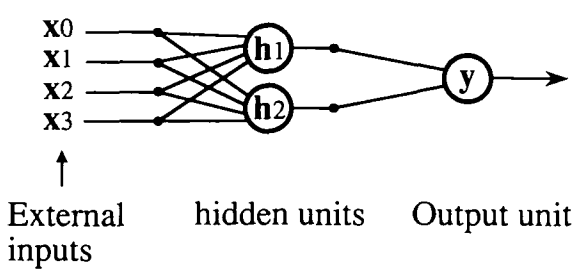


FIGURE 4. Layered feedforward network (for $n = 4$, $m = 2$, and one hidden layer).

The layered network is usually designed by BPL. In practice, however, there are many difficult tasks that are beyond the capability of BPL. Gradient descent algorithms often miss a goal by getting trapped in a local minimum of the error function. For BPL, the problem is partly due to poor sharing of error correction among hidden units. There is little cooperative and/or competitive relationships between them in the error-correcting process. This situation is improved in the maximization of Δ which includes such a relationship in the form of $\mathbf{P}_i$. Each $\mathbf{h}_i$ is modified to span a better subspace taking the other $\mathbf{h}_j$ into account.

Experiment 2. The experimental task was the same as in Experiment 1 except for the layered network structure. The initial conditions of w were given pseudorandom numbers generated from a uniform $(-1, 1)$ distribution. One $\mathbf{h}$ was modified with $\eta = 0.3$ every step. Every $\mathbf{h}$ was modified in turn until $\mathbf{y}$ satisfied C1: $\|\mathbf{z} - \mathbf{y}\| < 0.3$. Several trials for each functions were done from various initial conditions of $\mathbf{w}$. When the number of iteration steps ($\text{it}_{\max}$) reached 3,000, the trial was stopped.

The proposed method can obtain a minimum network for almost all functions. Table 3 shows the percentage of successful learning for each class of the functions, which are classified with respect to the ideally minimum number (m_i) of hidden units for constructing a layered TLN. For example, the only function with $m_i = 4$ is the 4-bit parity function. The m_i is a good indicator of the complexity of the functions, but it does not necessarily equal the minimum number of hidden units needed for the designed networks. There are two reasons for this, due to the difference between the designed networks and the TLN in the nonlinearity of the units. The output unit has a linear function so that the network needs one more nonlinear hidden unit for some functions. The logistic function of the hidden units produces an analog output, and so $\mathbf{h}$ can take a greater variety of directions than binary-component vectors of the TLN, which decreases the number of hidden units. Actually, some functions of $m_i = 3$ can be implemented by using fewer hidden units.

Table 3 also shows the results of similar experiments for BPL ($\eta = 0.1$ for $\delta\mathbf{w}$ of the hidden units and

TABLE 3
Percentage of Successful Learning in Experiment 2

		[%] Via the Proposed Method					[%] Via BPL				
mi	*Nf*	$m = 1$	=2	=3	=4	=5	$mb = 1$	=2	=3	=4	=5
=1	15	100	100	100	100	100	100	100	100	100	100
=2	133	0	72	95	100	100	0	67	96	100	100
=3	89	0	2	85	99	100	0	1	84	99	100
=4	1	0	0	0	52	94	0	0	0	36	74

Nf: number of representative Boolean functions.
mi: ideally minimum number of hidden units for layered TLN.
m, *mb*: number of hidden units designed by the proposed method or BPL.

$\eta = 0.01$ for $\delta \mathbf{w}_o$ of the output unit). Both methods give almost comparable results. For some functions, however, BPL is inferior to the proposed method. The parity function is one such difficult function for BPL, as shown in Table 4 for the 5- and 6-bit parity function.

One trial of successful convergence for $m = m_i$ required 470 iterations (2.5 seconds) for the proposed method and 780 iterations (0.6 seconds) for BPL, on average. The total computation time (5 trials $\times$ 237 functions for $m = m_i$) was 108 minutes for the proposed method and 30 minutes for BPL.

6. OPTIMIZATION FOR NONLINEAR OUTPUT UNITS

When the output unit has a nonlinear function to transform $\mathbf{y}$ into $g(\mathbf{y})$, the Euclidean norm $\|\mathbf{z} - \mathbf{y}\|_2$ is not a good criterion for optimizing $\mathbf{y}$. It is better to evaluate optimality with respect to $\|\mathbf{y}_o - \mathbf{y}\|_2$ where $\mathbf{y}_o$ is an objective state vector such that $\mathbf{z} = g(\mathbf{y}_o)$. When the function g is not invertible in practice, as is the case in a saturated region of a logistic function limited to $(-1, 1)$, $\mathbf{y}_o$ is neither known nor unique. It is necessary to search for an optimal $\mathbf{y}_o$ that satisfies $\mathbf{z} \approx g(\mathbf{y}_o)$.

To optimize $\mathbf{y}_o$ in such a case, it is better to modify $\mathbf{y}_o$ gradually so as to minimize the angle ϕ between $\mathbf{y}_o$ and $\mathbf{y}$, while $\mathbf{y}$ ($=\mathbf{P}'\mathbf{y}_o$) is modified simultaneously by changing $\mathbf{h}$ (i.e., $\mathbf{P}'$) to maximize Δ as mentioned above. The change in $\mathbf{y}_o$ with the gradient ascent algorithm to maximize $\cos^2\phi$ is given by

$$\delta y_{oi} \propto \frac{\partial}{\partial y_{oi}} \cos^2\phi, \quad \text{subject to} \quad \mathbf{z} \approx g(\mathbf{y}_o),$$

where $\cos^2\phi = (\mathbf{y}_o\mathbf{y})^2/(\mathbf{y}_o\mathbf{y}_o)(\mathbf{y}^T\mathbf{y})$. Therefore, $\delta\mathbf{y}_o$ is expressed by

$$\delta\mathbf{y}_o = \kappa\beta(\mathbf{y} - \beta\mathbf{y}_o)/(\mathbf{y}^T\mathbf{y}), \tag{15}$$

where $\beta = (\mathbf{y}_o^T\mathbf{y})/(\mathbf{y}_o^T\mathbf{y}_o)$, and κ is a positive constant. If $\|\mathbf{y}_o - \mathbf{y}\|^2$ were the objective to be minimized instead of ϕ, $\delta\mathbf{y}_o$ would be a simple form but would tend to make $\mathbf{y}_o$ and $\mathbf{y}$ converge to zero, because $\mathbf{y} = \mathbf{P}'\mathbf{y}_o$.

It should be noted that the total performance of the network should be evaluated with respect to $\|\mathbf{z} - g(\mathbf{y})\|$, not $\|\mathbf{y}_o - \mathbf{y}\|$. If g is the threshold function, it is better to define the norm as the maximum error (L_∞ norm) instead of the Euclidean norm (L_2 norm) (Burrascano & Lucci, 1990). The criterion of $\min\{\|\mathbf{z} - \mathbf{y}\|_\infty\} < 1$ is not only a sufficient condition but also a necessary condition for the threshold function. However, the L_∞ norm cannot be used for the objective function Δ, because $\|\mathbf{z} - \mathbf{y}\|_\infty$ for nonseparable $\mathbf{z}$ is always minimum at $\mathbf{y} = 0$, and never indicates the degree of nonseparability.

Experiment 3. The experimental task was the same as in Experiment 2 except for the nonlinearity of the output unit. The nonlinear function g was the sign function. Initially, $\mathbf{y}_o$ was set equal to $\mathbf{z}$, and then modified every step according to eqn (15). $\mathbf{y}_o$ can be changed in any direction provided that it satisfies $\min\{|y_{oi}|\} > 0.2$. The goal was defined by C3: $\min\{y_i z_i\} > 0.1$. The other constants were given as $\kappa = 0.4$, $\eta = 0.1$, and $\text{it}_{\max} = 5{,}000$.

The percentage of successful learning which reaches the goal is shown in Table 5. The results of the proposed method are better than those of Experiment 2. This is due to the nonlinearity of the output unit which in-

TABLE 4
Percentage of Successful Learning for 5- and 6-Bit Parity Functions

	[%] Via the Proposed Method				[%] Via BPL			
	$m = 5$	=6	=7	=8	$mb = 5$	=6	=7	=8
5-bit parity	94	100	100	100	67	95	100	100
6-bit parity	0	20	51	84	0	9	38	58

m, *mb*: number of hidden units.

TABLE 5
Percentage of Successful Learning in Experiment 3

		[%] Via the Proposed Method					[%] Via BPL				
mi	*Nf*	$m = 1$	=2	=3	=4	=5	$mb = 1$	=2	=3	=4	=5
=1	15	100	100	100	100	100	100	100	100	100	100
=2	133	0	82	99	100	100	0	61	84	89	89
=3	89	0	42	99	100	100	0	12	81	94	94
=4	1	0	0	17	85	98	0	0	0	42	69

Nf: number of representative Boolean functions.
mi: ideally minimum number of hidden units for layered TLN.
m, *mb*: number of hidden units designed by the proposed method or BPL.

creases the total nonlinearity of the network so as to perform more complex functions. On the other hand, the results of BPL ($\eta = 0.1$ and $it_{max} = 10{,}000$) are worse. The output unit's nonlinearity g, which is the same logistic function as the hidden units in this case, rather worsens the convergency for BPL, even for a redundant network having excess hidden units.

One trial of successful convergence for $m = m_i$ required 320 iterations (1.7 seconds) for the proposed method and 780 iterations (0.7 seconds) for BPL, on average.

7. DISCUSSION

The function of the hidden units is to produce state vectors by nonlinear transformation out of the input state space in order to span a larger state space. The expanded state space should be optimized so as to include an output state vector that satisfies Condition (3). The total function of the network is approximated by a linear combination of the state vectors that form a nonorthogonal basis of the expanded state space. In this optimization, the objective function Δ represents the performance of each hidden unit, so it can be used as a criterion for removing or adding hidden units.

Linear output is useful for constructing the network. Its advantages are that the total error of the network can be simply evaluated by $\|\mathbf{z} - \mathbf{y}\|$, and exactly analyzed for optimization by using practical theories of linear systems. Actually, the objective function Δ is derived exactly as shown in Section 4. On the other hand, the network using a linear output unit needs more hidden units than one using a nonlinear output unit, but there is not much difference in the necessary number of hidden units. For a fully connected feedforward network, for example, the nonlinear output unit can be regarded as the last hidden unit. It is easy to add a linear output unit to the network without changing anything else. The difference is only one hidden unit at most.

Fully connected feedforward neural networks have the highest capacity among feedforward networks. The hidden units can be optimized one by one in a relatively short time. This method has been shown to be very useful in designing minimum networks quickly. In principle, however, such a way will not necessarily form the best basis of the state space. For example, even if $L[\mathbf{X}\mathbf{h}_1]$ is the best for minimizing output errors, $\mathbf{h}_2$ produced from $L[\mathbf{X}\mathbf{h}_1]$ is not necessarily best for spanning $L[\mathbf{X}\mathbf{h}_1\mathbf{h}_2]$, which may not be better than $L[\mathbf{X}\mathbf{h}_1'\mathbf{h}_2']$ spanned by another $\mathbf{h}_2'$ produced from nonoptimal $L[\mathbf{X}\mathbf{h}_1']$. It is very difficult to optimize $\mathbf{h}_1$ to make the best combination with $\mathbf{h}_2$, because $\mathbf{h}_2$ varies greatly with $\mathbf{h}_1$.

For the layered networks, $\mathbf{h}$'s can be changed independently of each other, so $\mathbf{H}$ can be designed not only by the one-by-one addition of $\mathbf{h}$ which is optimized and fixed but also by the iterative modification of all $\mathbf{h}$ in turn. These two ways are distinguished by the definition of $\mathbf{H}_i$ as Types 1 and 2 shown in Section 5. The difference is whether one-sided dependence (Type 1) or interdependence (Type 2) of $\mathbf{h}$ vectors is taken into account in the optimization of $\mathbf{h}$. Type 1 is faster to compute than Type 2, but may not give a better solution in terms of the network size. Thus, Type 1 should be used for rough designing as the first step to estimate the upper limit of the necessary number of hidden units, and then Type 2 should follow to optimize the network by reducing the number of hidden units.

The key point of the Δ maximization is to introduce the interdependence of hidden units into optimality evaluation. The Δ maximization has a good effect on minimum network construction, but it takes more time to compute than the other methods (CCL for fully connected feedforward networks and BPL for layered networks). The computation time increases much more for larger scale problems. This is mainly due to redundant calculations for $\mathbf{P}_c$. However, the Δ can be reformulated by using inner products of $\mathbf{h}$ vectors, and separated into changing and unchanging parts for each step of an iterative calculation. Furthermore, if the sample data are presented to the network sequentially, the inner product of state vectors can be represented by the time integral of the product of states. Thus, it is possible to eliminate waste and reduce total computation time and memory space.

The nonlinear function f need not be the sigmoid function. Any nonlinear function can be used provided it can generate vectors out of the input state space to span a K-dimensional state space at most. For example, the sine function is more efficient for network minimization than the logistic function because of its higher nonlinearity. Products of inputs are useful for implementing the parity function, and periodic functions for shift-invariant pattern recognition. To minimize the size of network, it will be very important to consider what kind of nonlinearity is appropriate for an application.

Minimization of the number of modifiable parameters is closely related to generalization ability in connection with the principle of parsimony appearing in AIC. The generalization ability will be improved by real-time elimination of the excess hidden units which have lower performance in terms of Δ. However, generalization strongly depends on both the quantity and quality of sample data sets, so it should also be considered from a different point of view such as VC dimension (Baum & Haussler, 1989) and cross-validation (Morgan & Bourlard, 1990; Stone, 1987). The problem is concerned with characteristics of the row space of $[\mathbf{XH}]$ rather than the column space discussed here. This is a subject for future study.

8. CONCLUSIONS

This paper proposed the novel objective function, $\Delta = (\mathbf{z}^T\mathbf{P}_c\mathbf{h})^2/(\mathbf{h}^T\mathbf{P}_c\mathbf{h})$, for optimizing the additional hidden unit output state vector $\mathbf{h}$ for a desired output vector $\mathbf{z}$. The Δ is derived from the decrease in the least squared errors of output units caused by addition of the hidden unit to the network. In the state space, maximizing Δ is equal to minimizing the angle between $\mathbf{z}$ and the projection of $\mathbf{h}$ onto the orthogonal complement of the input state space. A nearly optimal $\mathbf{h}$ can be obtained using the gradient ascent algorithm. This optimization method is useful in constructing fully connected feedforward neural networks and for minimizing the size of layered networks.

REFERENCES

Akaho, S., & Amari, S. (1990). On the capacity of three-layer networks. *Proceedings of the International Joint Conference on Neural Networks*, San Diego, Vol. III, 1–6.

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, **AC-19**, 716–723.

Amari, S. (1967). Theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, **EC-16**, 299–307.

Ash, T. (1989). Dynamic node creation in backpropagation networks. *Proceedings of the International Joint Conference on Neural Networks*, Washington, D.C., June, Vol. II, 623. Piscataway, NJ: IEEE Service Center.

Baum, E. B. (1988). On the capabilities of multilayer perceptrons. *Journal of Complexity*, **4**, 193–215.

Baum, E. B., & Haussler, D. (1989). What size net gives valid generalization? *Neural Computation*, **1**, 151–160.

Burrascano, P., & Lucci, P. (1990). A learning rule in the Chebyshev norm for multilayer perceptrons. *Proceedings of the International Joint Conference on Neural Networks*, San Diego, Vol. III, 81–86. Piscataway, NJ: IEEE Service Center.

Deffuant, G. (1990). Neural units recruitment algorithm for generation of decision trees. *Proceedings of the International Joint Conference on Neural Networks*, San Diego, June, Vol. I, 637–642. Piscataway, NJ: IEEE Service Center.

Diederich, J. (1988). Connectionist recruitment learning. *Proceedings of the 8th European Conference on Artificial Intelligence*, Munich, August, 351–356.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: John Wiley & Sons.

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. In D. S. Touretzky (Ed.), *Advances in neural information processing systems 2* (pp. 524–532). San Mateo, CA: Morgan Kaufmann.

Fujita, O. (1990). A method for designing the internal representation of neural networks. *Proceedings of the International Joint Conference on Neural Networks*, San Diego, June, Vol. III, 149–154. Piscataway, NJ: IEEE Service Center.

Gallant, S. I. (1990). Perceptron-based learning algorithms. *IEEE Transactions on Neural Networks*, **1**, 179–191.

Golomb, S. W. (1959). On the classification of Boolean functions. *IRE Transactions on Circuit Theory*; **CT-6**, 176–186.

Hagiwara, M. (1990). Novel back propagation algorithm for reduction of hidden units and acceleration of convergence using artificial selection. *Proceedings of the International Joint Conference on Neural Networks*, San Diego, June, Vol. I, 625–630. Piscataway, NJ: IEEE Service Center.

Hirose, Y., Yamashita, K., & Hijiya, S. (1989). Back propagation method which varies the number of hidden units. *1989 Spring National Convention Record, the Institute of Electronics, Information and Communication Engineers*, Japan, **D-18, 7**, 18.

Ho, Y-C., & Kashyap, R. L. (1965). An algorithm for linear inequalities and its application. *IEEE Transactions on Electronic Computers*, **EC-14**, 683–688.

Hopcroft, J. E., & Mattson, R. L. (1965). Synthesis of minimal threshold logic networks. *IEEE Transactions on Electronic Computers*, **EC-14**, 552–560.

Kautz, W. H. (1961). The realization of symmetric switching functions with linear-input logical elements. *IRE Transactions on Electronic Computers*, **EC-10**, 371–378.

Kawahara, H., & Irino, T. (1988). Introduction to saturated projection algorithm for artificial neural network design. Research Reports, NTT Basic Research Laboratories, Musashino, Tokyo, Japan. (English summary of Japanese IEICE reports PRU88-54 and SP88-86, 1988).

Kurita, T. (1990). A method to determine the number of hidden units of three-layered neural networks by information criteria. *The Transactions of the Institute of Electronics, Information and Communication Engineers* (in Japanese), **J73-D-II**, 1872–1878.

Mezard, M., & Nadal, J.-P. (1989). Learning in feedforward layered networks: the tiling algorithm. *Journal of Physics*, **A22**, 2191–2203.

Morgan, N., & Bourlard, H. (1990). Generalization and parameter estimation in feedforward nets: Some experiments. In D. S. Touretzky (Ed.), *Advances in neural information processing systems 2* (pp. 630–637) San Mateo, CA: Morgan Kaufmann.

Ramacher, U., & Wesseling, M. (1989). A geometric approach to neural network design. *Proceedings of the International Joint Conference on Neural Networks*, Washington D.C., June, Vol. II, 147–152. Piscataway, NJ: IEEE Service Center.

Rujan, P., & Marchand, M. (1989). A geometric approach to learning in neural networks. *Proceedings of the International Joint Conference on Neural Networks*, Washington D.C., June, Vol. II, 105–110. Piscataway, NJ: IEEE Service Center.

Rumelhart, D. E., & McClelland, L. L. (1986). *Parallel distributed processing: Exploration in the microstructure of cognition.* Cambridge, MA: MIT Press.

Slepian, D. (1954). On the number of symmetry types of Boolean functions of n variables. *Canadian Journal of Mathematics,* **5,** 185–193.

Stone, M. (1987). Cross-validation: A review. *Mathematische Operationstorschung and Statistik, Series Statistics,* **9,** 127–139.

Webb, A. R., & Lowe, D. (1990). The optimized internal representation of multilayer classifier networks performs nonlinear discriminant analysis. *Neural Networks,* **3,** 367–375.

APPENDIX: PROJECTION MATRIX P′

Let **A** and **B** be square matrices that can be partitioned in the same form as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix}.$$

If **B** is the inverse matrix of **A**, the following equations hold.

$$\mathbf{B}_{11} = \mathbf{A}_{11}^{-1} + \mathbf{A}_{11}^{-1}\mathbf{A}_{12}\mathbf{B}_{22}\mathbf{A}_{21}\mathbf{A}_{11}^{-1},$$

$$\mathbf{B}_{12} = -\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\mathbf{B}_{22},$$

$$\mathbf{B}_{21} = -\mathbf{B}_{22}\mathbf{A}_{21}\mathbf{A}_{11}^{-1},$$

$$\mathbf{B}_{22} = (\mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12})^{-1}.$$

Thus, the inverse matrix **C** of $[\mathbf{Xh}]^T[\mathbf{Xh}]$, i.e.,

$$\begin{bmatrix} \mathbf{X}^T\mathbf{X} & \mathbf{X}^T\mathbf{h} \\ \mathbf{h}^T\mathbf{X} & \mathbf{h}^T\mathbf{h} \end{bmatrix},$$

is given by

$$\mathbf{C}_{11} = (\mathbf{X}^T\mathbf{X})^{-1} + (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{h}C\mathbf{h}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1},$$

$$\mathbf{C}_{12} = -(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{h}C,$$

$$\mathbf{C}_{21} = -C\mathbf{h}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1},$$

$$\mathbf{C}_{22} = C = (\mathbf{h}^T\mathbf{h} - \mathbf{h}^T\mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{h})^{-1} = (\mathbf{h}^T\mathbf{P}_c\mathbf{h})^{-1}.$$

Therefore,

$$\begin{aligned} \mathbf{P}' &= [\mathbf{Xh}]([\mathbf{Xh}]^T[\mathbf{Xh}])^{-1}[\mathbf{Xh}]^T \\ &= \mathbf{X}\mathbf{C}_{11}\mathbf{X}^T + \mathbf{X}\mathbf{C}_{12}\mathbf{h}^T + \mathbf{h}\mathbf{C}_{21}\mathbf{X}^T + \mathbf{h}\mathbf{C}_{22}\mathbf{h}^T \\ &= \mathbf{P} + \mathbf{P}\mathbf{h}C\mathbf{h}^T\mathbf{P} - \mathbf{P}\mathbf{h}C\mathbf{h}^T - \mathbf{h}C\mathbf{h}^T\mathbf{P} + \mathbf{h}C\mathbf{h}^T \\ &= \mathbf{P} + \mathbf{P}_c\mathbf{h}C\mathbf{h}^T\mathbf{P}_c. \end{aligned}$$