*IEICE Transactions on Information and Systems, vol.E85-D, no.9, pp.1433–1442, Sep. 2002.*

# Incremental Construction of Projection Generalizing Neural Networks

Masashi Sugiyama Hidemitsu Ogawa

Department of Computer Science,
Graduate School of Information Science and Engineering,
Tokyo Institute of Technology.

2-12-1, O-okayama, Meguro-ku, Tokyo, 152-8552, Japan.

sugi@og.cs.titech.ac.jp
http://ogawa-www.cs.titech.ac.jp/~sugi/

**Abstract**

In many practical situations in NN learning, training examples tend to be supplied one by one. In such situations, incremental learning seems more natural than batch learning in view of the learning methods of human beings. In this paper, we propose an incremental learning method in neural networks under the projection learning criterion. Although projection learning is a linear learning method, achieving the above goal is not straightforward since it involves redundant expressions of functions with over-complete bases, which is essentially related to pseudo biorthogonal bases (or frames). The proposed method provides exactly the same learning result as that obtained by batch learning. It is theoretically shown that the proposed method is more efficient in computation than batch learning.

**Keywords**

generalization capability, incremental learning, pseudo biorthogonal basis (PBOB), projection learning, incremental projection learning, projection generalizing neural network.

## 1 Introduction

The purpose of learning in neural networks (NNs) is estimating an unknown input-output rule from a finite number of training examples. If the rule is successfully identified, then it is possible to estimate output values corresponding to novel input points. This ability is called the *generalization capability.*

If we pay attention to the input-output relation of a NN, it can be regarded as a function. Based on this interpretation, the NN learning is formulated as an inverse problem from the functional analytic point of view [15, 16]. This framework divides the NN learning into two stages. In the first stage, function approximation is performed on the basis of a learning criterion independent of the architecture of NNs. So far, various learning criteria have been proposed depending on the purpose, e.g., *least mean squares learning*, *projection learning* [14], *Wiener learning* [18], *parametric projection learning* [20], and *regularization learning* [11]. Also, methods for calculating learning result functions by using all given training examples in a batch manner have been devised in the references [14, 18, 20, 11]. In the second stage, a NN that represents the learning result function is constructed. A general construction method of NNs has been proposed [16], which enables us to perform batch learning in NNs on the basis of each learning criterion.

In many practical situations in NN learning, however, training examples tend to be supplied one by one. In such situations, *incremental learning* seems more natural than batch learning in view of the learning methods of human beings. Although many incremental learning methods devised so far are more efficient than batch learning in computation, they do not generally provide as good generalization capability as batch learning [21, 8, 9, 33, 32, 29, 24]. An incremental learning method which asymptotically provides the same generalization capability as that obtained by batch learning was proposed [3]. Even with this method, however, learning results do not agree with those obtained by batch learning in the non-asymptotic case, causing a crucial problem since the number of training examples is always finite in practice. A method of incremental projection learning corresponding to the function approximation stage of the above mentioned framework was given [26]. This method provides exactly the same learning result as that obtained by batch projection learning with finite training examples. Properties of incremental projection learning have been investigated in detail in the reference [27].

The aim of this paper is to give an incremental construction method of NNs under the projection learning criterion, which enables us to perform incremental projection learning in NNs. We are dealing with linear systems so that incremental projection learning in NNs seems rather straightforward. However, this is not true since it involves redundant expressions of functions with over-complete bases, which is essentially related to the *pseudo biorthogonal bases* [13, 17] (or *frames* [4, 6]).

This paper is organized as follows. In Section 2, the NN learning problem is formulated and the definition of projection learning is described. Section 3 reviews a method of batch projection learning in NNs. Essentially, Sections 2 and 3 are reviews of the previous works, so readers may skip over these sections and go straightly to Section 4. Section 4 gives a method of incremental projection learning in NNs. In Section 5, the proposed method is analyzed regarding the condition of adding new hidden units. Finally, Section 6 is devoted to computer simulations for experimentally investigating the effectiveness of the proposed method.

## 2 Formulation of NN learning problem

In this section, the NN learning problem is formulated following the reference [15, 16], and then the definition of projection learning [14] is reviewed. This section is essentially the review of the above papers so readers who are interested in the main contents may skip over this section.

### 2.1 NN learning as an inverse problem

Let us consider a learning problem of a three-layer feedforward NN with the numbers of input and output units being $L$ and 1, respectively. The relationship between input $x = (\eta_1, \ldots, \eta_L)^\top$ and output $y$ of the NN is expressed by using a function $f_0(x)$ of $L$ variables as

$$y = f_0(x). \tag{1}$$

The NN learning problem is to obtain the optimal approximation to a target function $f$ from a set of $m$ training examples made up of input signals $x_i \in \mathbf{R}^L$ and corresponding output signals $y_i \in \mathbf{C}$:

$$\{(x_i, y_i) \mid y_i = f(x_i) + n_i : i = 1, 2, \ldots, m\}, \tag{2}$$

where $y_i$ is degraded by additive noise $n_i$. Basically, we assume that the noise covariance matrix is known through this paper. However, as shown in Section 4.1, this assumption is not needed if the noise covariance matrix is known to be proportional to the identity matrix.

In many NN learning methods devised so far, learning algorithms are built upon a certain architecture of NNs, i.e., a fixed number of hidden units, each with a prespecified *sigmoidal* or *radial basis functions*. However, the restrictions sometimes prevent us from obtaining the optimal approximation. Therefore, we may divide our NN learning problem into two stages. Function approximation from given training examples is performed in the first stage, and a NN which represents the approximated function is constructed in the second stage.

To begin with, we formulate the function approximation problem corresponding to the first stage. Let $n^{(m)}$ and $y^{(m)}$ be $m$-dimensional vectors with the $i$-th elements being $n_i$ and $y_i$, respectively. $y^{(m)}$ is called a *sample value vector*, and a space to which $y^{(m)}$ belongs is called a *sample value space*. In this paper, the target function $f$ is assumed to belong to a reproducing kernel Hilbert space $H$ [2]. (see also [30, 5]). Let us denote the reproducing kernel of $H$ by $K(x, x')$. If a function $\psi_i(x)$ is defined as

$$\psi_i(x) = K(x, x_i), \tag{3}$$

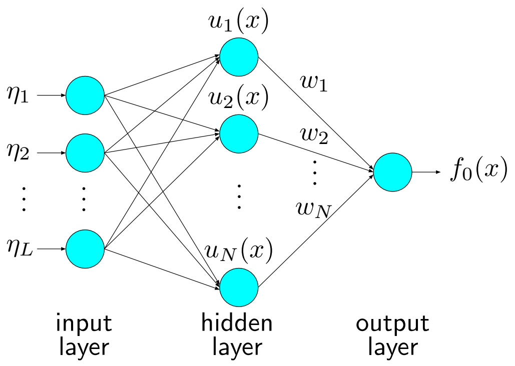


Figure 1: The structure of a neural network.

then the value of $f$ at a sample point $x_i$ is expressed as

$$f(x_i) = \langle f, \psi_i \rangle. \tag{4}$$

For this reason, $\psi_i$ is called a *sampling function*. Let $A_m$ be an operator which maps $f$ to an $m$-dimensional vector with the $i$-th element being $f(x_i)$. We call $A_m$ a *sampling operator*. Then the relationship between $f$ and $y^{(m)}$ can be expressed as

$$y^{(m)} = A_m f + n^{(m)}. \tag{5}$$

Note that $A_m$ is always a linear operator even when we are concerned with a non-linear function $f$. Indeed, $A_m$ can be expressed by using the *Neumann-Schatten product*[1] as

$$A_m = \sum_{i=1}^{m} \left( e_i^{(m)} \otimes \overline{\psi_i} \right), \tag{6}$$

where $e_i^{(m)}$ is the $i$-th vector of the so-called standard basis in $\mathbf{C}^m$. Let $f_m$ be a learning result function obtained from $m$ training examples and $X_m$ be an operator which maps $y^{(m)}$ to $f_m$:

$$f_m = X_m y^{(m)}. \tag{7}$$

$X_m$ is called a *learning operator*. Then the first stage of the NN learning problem can be reformulated as an inverse problem of obtaining $X_m$ that provides the best approximation $f_m$ to $f$ under a certain criterion.

Now we go on to the second stage, i.e., the construction of a NN which represents $f_m$. In the second stage, the number $N$ of hidden units, basis functions $\{u_j(x)\}_{j=1}^N$, and

[1]For any fixed $g$ in a Hilbert space $H_1$ and any fixed $f$ in a Hilbert space $H_2$, the *Neumann-Schatten product* $(f \otimes \overline{g})$ is an operator from $H_1$ to $H_2$ defined by using any $h \in H_1$ as follows [25]:

$$(f \otimes \overline{g})h = \langle h, g \rangle f.$$

weights $\{w_j\}_{j=1}^N$ on hidden-output connections are determined (Fig.1). In conventional neural networks, the following basis function is commonly used:

$$u_j(x) = \sigma(\sum_{k=1}^{L} w_{kj}\eta_k), \tag{8}$$

where $\sigma(\cdot)$ is a sigmoidal activation function and $w_{kj}$ is a weight on the connection between the $k$-th input unit and the $j$-th hidden unit. In this paper, $\{u_j(x)\}_{j=1}^N$ is given by e.g. the reproducing kernel of $H$ (see [5] for convenient reproducing kernel Hilbert spaces).

## 2.2 Projection learning

As mentioned above, the function approximation problem is formulated as an inverse problem. Since image and signal restoration problems discussed in the references [14, 19] are also formulated as the same form of inverse problems, the optimal image restoration filters devised in these papers can be applied to the function approximation problem discussed in this paper. We adopt the projection learning criterion. Let $E_n$, $A_m^*$, $\mathcal{R}(A_m^*)$, and $P_{\mathcal{R}(A_m^*)}$ be the ensemble average over the noise, the adjoint operator of $A_m$, the range of $A_m^*$, and the orthogonal projection operator onto $\mathcal{R}(A_m^*)$, respectively. Then projection learning is defined as follows.

**Definition 1 (Projection learning)** *[14] An operator $X_m$ is called the projection learning operator if $X_m$ minimizes the functional*

$$J_P[X_m] = E_n\|X_m n^{(m)}\|^2 \tag{9}$$

*under the constraint*

$$X_m A_m = P_{\mathcal{R}(A_m^*)}. \tag{10}$$

Since the learning result function $f_m$ is searched in $\mathcal{R}(A_m^*)$, this space is called the *approximation space* for $f_m$. From Eqs.(7) and (5), the learning result function $f_m$ can be decomposed as

$$f_m = X_m A_m f + X_m n^{(m)}. \tag{11}$$

The first and second terms of Eq.(11) are called the *signal* and *noise components* of $f_m$, respectively. The projection learning criterion requires the signal component to coincide with the orthogonal projection of $f$ onto $\mathcal{R}(A_m^*)$ and the noise component to minimize its variance.

It has been shown that learning results obtained by projection learning are invariant under the inner product in a sample value space [31]. Hence, the Euclidean inner product is adopted without loss of generality.

## 3 Batch projection learning in NNs

In this section, we review a method of batch projection learning in NNs, which forms a basis for devising incremental learning techniques in the following sections. Therefore, readers who are interested in the main contents may also skip over this section. We first review batch projection learning [14] that corresponds to the function approximation stage of our framework. Then we go on to the second stage and review the method of constructing NNs in a batch manner [16]. Here, the concept of pseudo orthogonal bases plays an essential role [13, 17].

Let $I_m$ be the identity matrix on $\mathbf{C}^m$ and $A_m^\dagger$ be the *Moore-Penrose generalized inverse* of $A_m$ [1]. Then the following proposition holds.

**Proposition 1** *[14] A general form of the projection learning operator $A_m^{(P)}$ is expressed as*

$$A_m^{(P)} = V_m^\dagger A_m^* U_m^\dagger + Y_m(I_m - U_m U_m^\dagger), \tag{12}$$

*where*

$$\begin{aligned} Q_m &= E_n\left(n^{(m)} \otimes \overline{n^{(m)}}\right), & (13)\\ U_m &= A_m A_m^* + Q_m, & (14)\\ V_m &= A_m^* U_m^\dagger A_m, & (15)\end{aligned}$$

*and $Y_m$ is an arbitrary operator from $\mathbf{C}^m$ to $H$.*

By using Eqs.(12) and (7), we can calculate a projection learning result function $f_m$ from $m$ given training examples in a batch manner. This method is called *batch projection learning*. Note that $f_m$ is uniquely determined in spite of the non-uniqueness of the projection learning operator caused by $Y_m$.

A NN that represents a projection learning result function is called a *projection generalizing NN* (PGNN). The construction of NNs is mathematically equivalent to an expansion of the learning result function $f_m(x)$ by basis functions $\{u_j(x)\}_{j=1}^N$, where $N$ is the number of hidden units and $u_j(x)$ is an input-output function of the $j$-th hidden unit (see Fig.1). When $\{u_j\}_{j=1}^N$ is an orthonormal basis (ONB) in $H$, the NN construction problem becomes an expansion of $f_m$ by ONBs. When $N$ is larger than the dimension of $H$, this problem becomes an expansion by *pseudo biorthogonal bases*, which is an extension of ONBs defined as follows.

**Definition 2 (Pseudo biorthogonal bases)** *[13, 17]*
*Let $\{u_j, u_j^*\}_{j=1}^N$ be a set of $2N$ ($N \geq \mu$) elements in a $\mu$-dimensional Hilbert space $H$. If any $f$ in $H$ can be expressed as*

$$f = \sum_{j=1}^{N} \langle f, u_j^* \rangle u_j, \tag{16}$$

*then $\{u_j, u_j^*\}_{j=1}^N$ is called a pseudo biorthogonal basis (PBOB) in $H$.*

A PBOB is also referred to as a *frame* in the wavelet literature [4, 6]. When $\{u_j, u_j^*\}_{j=1}^N$ is a PBOB, $\{u_j^*\}_{j=1}^N$ is called a *dual sequence* to $\{u_j\}_{j=1}^N$. If $N$ is equal to the dimension of $H$ and $u_j^* = u_j$ for all $j$, then a PBOB is reduced to an ONB. Hence, the concept of PBOBs is natural extension of ONBs. Indeed, PBOBs inherit many useful properties from ONBs, including Parseval's equalities [17]. A dual sequence $\{u_j^*\}_{j=1}^N$ can be calculated from $\{u_j\}_{j=1}^N$ as follows.

**Proposition 2 (Batch calculation of PBOBs)** *[13, 17] Let $G_N$ and $W_N$ be defined as*

$$G_N = \sum_{j=1}^{N} \left( e_j^{(N)} \otimes \overline{u_j} \right), \tag{17}$$

$$W_N = (G_N^\dagger)^* + (I_N - G_N G_N^\dagger) Z_N, \tag{18}$$

*where $Z_N$ is an arbitrary operator from $H$ to $\mathbf{C}^N$. If we let*

$$u_j^* = W_N^* e_j^{(N)}, \tag{19}$$

*then $\{u_j, u_j^*\}_{j=1}^N$ forms a PBOB in $H$.*

Based on the concept of PBOBs, we shall review a batch construction method of PGNNs, which we refer to as BPGNN. Let $w^{(N)}$ be an $N$-dimensional vector with the $j$-th element being the weight $w_j$ on the connection to the $j$-th hidden unit (see Fig.1). $w^{(N)}$ is called a *weight vector.* Then BPGNN is given as follows.

**Proposition 3 (BPGNN)** *[16] A NN that satisfies the following conditions is a PGNN, and all PGNNs can be constructed by this method.*

1. *The number $N$ of hidden units:*

$$N \geq rank(A_m^*). \tag{20}$$

2. *Basis functions $\{u_j(x)\}_{j=1}^N$ of hidden units:*

$$\mathcal{L}\left(\{u_j\}_{j=1}^N\right) \supset \mathcal{R}(A_m^*), \tag{21}$$

*where $\mathcal{L}\left(\{u_j\}_{j=1}^N\right)$ denotes the subspace spanned by $\{u_j\}_{j=1}^N$.*

3. *Weights $\{w_j\}_{j=1}^N$ on hidden-output connections:*

$$w^{(N)} = W_N A_m^{(P)} y^{(m)}. \tag{22}$$

Eq.(21) implies that any basis functions $\{u_j(x)\}_{j=1}^N$ can be used as long as they satisfy Eqs.(20) and (21). In the so-called *back propagation* algorithm [23], it is said that the generalization capability becomes poor if the number of hidden units is too large [10]. For this reason, *pruning* algorithms are often used [22, 33].

However, one of the features which distinguish NNs from other learning machines is their robustness achieved by employing redundant hidden units. Therefore, the existence of redundant units is the nature of NNs. From this viewpoint, developing a learning theory for NNs with redundant hidden units is essential. This can be achieved by the framework described in Section 2.1, i.e., the function approximation stage is independent of the architecture of NNs. Indeed, Proposition 3 showed that the number $N$ of hidden units can be as large as desired in PGNNs without losing the generalization capability. By utilizing the redundancy of PGNN, a robust construction method of PGNNs in a batch manner was given [12] (see also [7]). NNs constructed by this method are specifically resistant to noise on the output of hidden units and connection faults.

In the next section, we give an incremental learning method in NNs with redundant units.

## 4 Incremental projection learning in NNs

So far, we formulated supervised learning, and reviewed a construction method of projection generalizing neural networks in a batch manner. In this section, we consider an incremental setting. First, we briefly review incremental projection learning [26] that corresponds to the first stage of our framework (see Section 2.1). Based on these preliminaries, we give a method of incremental projection learning in NNs in Section 4.2, which is our main contribution.

### 4.1 Incremental projection learning

Let us consider the case where a new training example $(x_{m+1}, y_{m+1})$ is added after a PGNN has been constructed from $m$ training examples. This case is fairly common in practical situations. For example, when we buy a character recognition equipment, it has been trained with general training examples. Then training examples created by each user are added for improving the recognition property.

Let noise characteristics of the additional training example $(x_{m+1}, y_{m+1})$ be

$$\begin{aligned} q_{m+1} &= E_n(\overline{n_{m+1}} n^{(m)}), && (23) \\ \sigma_{m+1} &= E_n |n_{m+1}|^2, && (24) \end{aligned}$$

where $\overline{n_{m+1}}$ denotes the complex conjugate of $n_{m+1}$.

Let $\mathcal{N}(A_m)$ be the null space of $A_m$ and $P_{\mathcal{N}(A_m)}$ is the orthogonal projection onto $\mathcal{N}(A_m)$. Let us define the following notation.

Matrix:

$$\Gamma_{m+1} = \sum_{i=1}^{m} \left( e_i^{(m+1)} \otimes \overline{e_i^{(m)}} \right). \tag{25}$$

Vectors:

$$s_{m+1} = A_m \psi_{m+1} + q_{m+1}, \tag{26}$$

$$t_{m+1} = U_m^{\dagger} s_{m+1}. \tag{27}$$

Scalars:

$$\alpha_{m+1} = \psi_{m+1}(x_{m+1}) + \sigma_{m+1} - \langle t_{m+1}, s_{m+1} \rangle, \tag{28}$$

$$\beta_{m+1} = y_{m+1} - f_m(x_{m+1}) - \langle y^{(m)} - A_m f_m, t_{m+1} \rangle. \tag{29}$$

Functions:

$$\tilde{\psi}_{m+1} = P_{\mathcal{N}(A_m)} \psi_{m+1}, \tag{30}$$

$$\xi_{m+1} = \psi_{m+1} - A_m^* t_{m+1}, \tag{31}$$

$$\tilde{\xi}_{m+1} = V_m^{\dagger} \xi_{m+1}. \tag{32}$$

The additional training examples which yield $\xi_{m+1} = 0$ can be rejected since they have no effect on learning results [27]. In the following discussion, we focus on the case where $\xi_{m+1} \neq 0$. Corresponding to the function approximation stage of the framework described in Section 2.1, a method of incremental projection learning (IPL) is given as follows.

**Proposition 4 (IPL)** *[26] When $\xi_{m+1}$ in Eq.(31) is not zero, a posterior projection learning result function $f_{m+1}$ can be obtained by using prior results $f_m$, $A_m$, $U_m^{\dagger}$, $V_m^{\dagger}$, and $y^{(m)}$ as*

$$f_{m+1} = \begin{cases} f_m + \beta_{m+1} \zeta_{m+1}^{(a)} & \text{if } \psi_{m+1} \in \mathcal{R}(A_m^*), \\ f_m + \beta_{m+1} \zeta_{m+1}^{(b)} & \text{if } \psi_{m+1} \notin \mathcal{R}(A_m^*), \end{cases} \tag{33}$$

*where $\zeta_{m+1}^{(a)}$ and $\zeta_{m+1}^{(b)}$ are defined as*

$$\zeta_{m+1}^{(a)} = \frac{\tilde{\xi}_{m+1}}{\alpha_{m+1} + \langle \tilde{\xi}_{m+1}, \xi_{m+1} \rangle}, \tag{34}$$

$$\zeta_{m+1}^{(b)} = \frac{\tilde{\psi}_{m+1}}{\tilde{\psi}_{m+1}(x_{m+1})}. \tag{35}$$

Note that $f_{m+1}$ obtained by Proposition 4 exactly coincides with the learning result function obtained by batch projection learning with $\{(x_i, y_i)\}_{i=1}^{m+1}$. The condition $\psi_{m+1} \in \mathcal{R}(A_m^*)$ means that $\psi_{m+1}$ belongs to the subspace spanned by $\{\psi_i\}_{i=1}^{m}$.

Let $V'_m$ and $\beta'_{m+1}$ be defined as

$$V'_m = A_m^* Q_m^{\dagger} A_m, \tag{36}$$

$$\beta'_{m+1} = y_{m+1} - f_m(x_{m+1}). \tag{37}$$

Then in a special case, IPL is reduced to a simpler expression as follows.

**Proposition 5** *[27] If the noise correlation matrix is positive definite and diagonal, i.e.,*

$$Q_{m+1} = diag(\sigma_1, \sigma_2, \ldots, \sigma_{m+1}), \tag{38}$$

*where $\sigma_i > 0$ for all $i$, then a posterior projection learning result function $f_{m+1}$ can be obtained by using prior results $f_m$ and $V_m'^{\dagger}$ as*

$$f_{m+1} = \begin{cases} f_m + \beta'_{m+1}\zeta^{(a)\prime}_{m+1} & \text{if } \psi_{m+1} \in \mathcal{R}(A_m^*), \\ f_m + \beta'_{m+1}\zeta^{(b)\prime}_{m+1} & \text{if } \psi_{m+1} \notin \mathcal{R}(A_m^*), \end{cases} \tag{39}$$

*where $\zeta^{(a)\prime}_{m+1}$ is defined as*

$$\zeta^{(a)\prime}_{m+1} = \frac{V_m'^{\dagger}\psi_{m+1}}{\sigma_{m+1} + \langle V_m'^{\dagger}\psi_{m+1}, \psi_{m+1}\rangle}, \tag{40}$$

*and $\zeta^{(b)\prime}_{m+1}$ is equal to $\zeta^{(b)}_{m+1}$ defined by Eq.(35).*

In the following discussion, $\beta_{m+1}$, $\zeta^{(a)}_{m+1}$, and $\zeta^{(b)}_{m+1}$ can be replaced with $\beta'_{m+1}$, $\zeta^{(a)\prime}_{m+1}$, and $\zeta^{(b)\prime}_{m+1}$ if the noise correlation matrix is in the form of Eq.(38) with $\sigma_i > 0$ for all $i$. Note that if $\sigma_1 = \sigma_2 = \cdots = \sigma_{m+1} = \sigma$, the value of $\sigma$ is not required since it is canceled out in Eq.(40).

## 4.2 Incremental construction of PGNN

Based on IPL, we shall give an incremental construction method of PGNN that corresponds to the second stage of our framework. We refer to the incremental construction method as IPGNN.

As mentioned in Section 3, the concept of PBOBs plays an essential role in the construction of NNs. First, we give an incremental calculation method of PBOBs, which enable us to calculate a dual sequence $\{u_j^*\}_{j=1}^{N+1}$ to $\{u_j\}_{j=1}^{N+1}$ after a PBOB $\{u_j, u_j^*\}_{j=1}^{N}$ has been obtained:

**Theorem 1 (Incremental calculation of PBOBs)** *If $u_{N+1} \notin \mathcal{L}\left(\{u_j\}_{j=1}^{N}\right)$, then $\{u_j^*\}_{j=1}^{N+1}$ can be obtained by using $\{u_j, u_j^*\}_{j=1}^{N}$ as*

$$\begin{aligned} u_j^* &\leftarrow u_j^* - \langle u_{N+1}, u_j^*\rangle + Y_N P_{\mathcal{N}(G_N^*)} e_j^{(N)} \\ &\qquad\qquad \text{for } 1 \le j \le N, & (41) \\ u_{N+1}^* &\leftarrow \tilde{u}_{N+1}, & (42) \end{aligned}$$

*where*

$$\tilde{u}_{N+1} = \frac{P_{\mathcal{N}(G_N)} u_{N+1}}{\|P_{\mathcal{N}(G_N)} u_{N+1}\|^2}, \tag{43}$$

*and $Y_N$ is an arbitrary operator from $\mathbf{C}^N$ to $H$.*

A proof of Theorem 1 is given in A. Let us measure the computational complexity by the number of scalar multiplications. If the batch method shown by Proposition 2 is used for obtaining $\{u_j^*\}_{j=1}^{N+1}$, the computational complexity is $\mathcal{O}(N^3)$. In contrast, the computational complexity required for Theorem 1 is reduced to $\mathcal{O}(N^2)$.

Combining Proposition 4 and Theorem 1, we have the following incremental construction method of PGNNs.

**Theorem 2 (IPGNN)** *A posterior PGNN can be incrementally constructed as follows.*

1. *When $\psi_{m+1} \in \mathcal{R}(A_m^*)$, the existing weights $\{w_j\}_{j=1}^N$ are modified as*

$$w^{(N)} \leftarrow w^{(N)} + \beta_{m+1} W_N \zeta_{m+1}^{(a)} + z, \tag{44}$$

*where $z$ is an arbitrary vector in $\mathcal{N}(G_N^*)$.*

2. *When $\psi_{m+1} \notin \mathcal{R}(A_m^*)$ and $\psi_{m+1} \in \mathcal{L}\left(\{u_j\}_{j=1}^N\right)$, the existing weights $\{w_j\}_{j=1}^N$ are modified as*

$$w^{(N)} \leftarrow w^{(N)} + \beta_{m+1} W_N \zeta_{m+1}^{(b)} + z. \tag{45}$$

3. *When $\psi_{m+1} \notin \mathcal{R}(A_m^*)$ and $\psi_{m+1} \notin \mathcal{L}\left(\{u_j\}_{j=1}^N\right)$, a novel hidden unit with the input-output function $u_{N+1}$ being a function such that $u_{N+1} \in \mathcal{R}(A_{m+1}^*)$ and $u_{N+1} \notin \mathcal{R}(A_m^*)$ is added. The weight $w_{N+1}$ on the connection to the novel unit is determined as*

$$w_{N+1} = \beta_{m+1} \langle \zeta_{m+1}^{(b)}, \tilde{u}_{N+1} \rangle. \tag{46}$$

*The existing weights $\{w_j\}_{j=1}^N$ are modified as*

$$\begin{aligned} w^{(N)} \ \leftarrow\ & w^{(N)} + \beta_{m+1} W_N \\ & \times (\zeta_{m+1}^{(b)} - \langle \zeta_{m+1}^{(b)}, \tilde{u}_{N+1} \rangle u_{N+1}) + z. \end{aligned} \tag{47}$$

A proof of Theorem 2 is given in B. PGNNs constructed by IPGNN (Theorem 2) express exactly the same functions as those constructed by BPGNN (Proposition 3). Theorem 2 states that a posterior learning result function $f_{m+1}$ can be represented without adding any novel units when 1. $\psi_{m+1} \in \mathcal{R}(A_m^*)$, or 2. $\psi_{m+1} \notin \mathcal{R}(A_m^*)$ and $\psi_{m+1} \in \mathcal{L}\left(\{u_j\}_{j=1}^N\right)$. Otherwise, the addition of a novel unit is indispensable. Eqs.(44), (45), and (47) imply that there exists freedom of determining the weights. By utilizing the freedom, we will give robust construction method of NNs in our future work.

Now we investigate the computational complexity required for BPGNN and IPGNN when $(x_{m+1}, y_{m+1})$ is added after a PGNN trained with $\{(x_i, y_i)\}_{i=1}^m$ has been obtained. When we use BPGNN for obtaining a posterior PGNN, the computational complexity is $\mathcal{O}(m^3 + N^3)$. In contrast, the computational complexity required for IPGNN to add the $(m+1)$-st training example to a prior NN is $\mathcal{O}(m^2 + N^2)$. Therefore, it is theoretically confirmed that the computational complexity required for IPGNN is less than that required for BPGNN. Note that the memory required for BPGNN and IPGNN is both $\mathcal{O}(m^2 + N^2)$ (Table 1). IPGNN can be calculated without the gradient descent method,

Table 1: The computational complexity required for BPGNN and IPGNN to add the $(m+1)$-st training example. $m$ and $N$ denote the numbers of training examples and hidden units, respectively.

| | Computational complexity | Memory |
|---|---|---|
| BPGNN | $\mathcal{O}(m^3+N^3)$ | $\mathcal{O}(m^2+N^2)$ |
| IPGNN | $\mathcal{O}(m^2+N^2)$ | $\mathcal{O}(m^2+N^2)$ |

which requires much computation until convergence. Hence, IPGNN is expected to be practically efficient in computation compared with gradient-descent-based methods.

When $N$ is equal to the dimension of $\mathcal{R}(A_m^*)$, i.e., equality in Eq.(20) holds, Theorem 2 is reduced to the following simpler expressions.

**Corollary 1** *When the number $N$ of hidden units in the prior PGNN is equal to the dimension of $\mathcal{R}(A_m^*)$, the posterior PGNN can be incrementally constructed as follows.*

1. *When $\psi_{m+1} \in \mathcal{R}(A_m^*)$, the existing weights $\{w_j\}_{j=1}^N$ are modified as*

$$w^{(N)} \leftarrow w^{(N)} + \beta_{m+1}(G_N^\dagger)^* \zeta_{m+1}^{(a)}. \tag{48}$$

2. *When $\psi_{m+1} \notin \mathcal{R}(A_m^*)$, the novel hidden unit with the input-output function $u_{N+1}(x)$ being $\tilde{\psi}_{m+1}(x)$ is added, and the weight $w_{N+1}$ on the connection to the novel unit is determined as*

$$w_{N+1} = \frac{\beta_{m+1}}{\tilde{\psi}_{m+1}(x_{m+1})}. \tag{49}$$

*The existing weights are used without any modifications.*

**Corollary 2** *When the number $N$ of hidden units in the prior PGNN is equal to the dimension of $\mathcal{R}(A_m^*)$, the posterior PGNN can be incrementally constructed as follows.*

1. *When $\psi_{m+1} \in \mathcal{R}(A_m^*)$, the existing weights $\{w_j\}_{j=1}^N$ are modified as Eq.(48).*

2. *When $\psi_{m+1} \notin \mathcal{R}(A_m^*)$, the novel hidden unit with the input-output function $u_{N+1}(x)$ being $\psi_{m+1}(x)$ is added, and the weight $w_{N+1}$ is determined by Eq.(49). The existing weights $\{w_j\}_{j=1}^N$ are modified as*

$$w^{(N)} \leftarrow w^{(N)} - \beta_{m+1}(G_N^\dagger)^* \zeta_{m+1}^{(b)}. \tag{50}$$

Corollary 1 states that the existing weights need not to be modified when $\psi_{m+1} \notin \mathcal{R}(A_m^*)$. This property is useful because it contributes to reducing mechanical trouble. However, Corollary 1 generally requires comparatively complicated hidden units which are capable of representing arbitrary functions in $H$. In contrast, Corollary 2 requires simple hidden units which are capable of representing only $K(x,x')$, the reproducing kernel in $H$. When a novel unit is added to NNs, $x_{m+1}$ is assigned to $x'$. This feature is preferable from the engineering point of view.

## 5 Discussion

In this section, we compare the proposed incremental learning method with the *resource allocating network* (RAN) [21] regarding the condition of adding new hidden units.

RAN is an incremental learning algorithm in radial basis function (RBF) networks. In RAN, the necessity of adding a novel hidden unit is judged by the following *novelty criteria*:

$$|x_{m+1} - (\text{center of nearest RBF})| > \delta_x, \tag{51}$$
$$|y_{m+1} - f_m(x_{m+1})| > \delta_y, \tag{52}$$

where $\delta_x$ and $\delta_y$ are appropriately determined thresholds. If the additional training example $(x_{m+1}, y_{m+1})$ satisfies both of the novelty criteria, then a new novel hidden unit with the center and height being $x_{m+1}$ and $(y_{m+1} - f_m(x_{m+1}))$, respectively, is added. Otherwise, any new hidden units are not employed and the existing parameters are modified by the gradient descent method so that

$$(y_{m+1} - f_m(x_{m+1}))^2 \tag{53}$$

is minimized.

As mentioned in Section 3, the robustness of NNs can be achieved by employing redundant units. On the other hand, the number of units should be kept as small as possible in order to reduce the computational complexity. The novelty criteria work well for balancing this trade-off. However, the novelty criteria do not always provide the required number of hidden units for obtaining the same learning result as batch learning, that minimizes

$$\sum_{i=1}^{m+1} (y_i - f_{m+1}(x_i))^2. \tag{54}$$

In contrast, the necessity of adding novel hidden units in IPGNN is judged by the criterion whether the prior NN is capable of expressing functions in the posterior approximation space. Eq.(20) shows the minimum number of hidden units required for acquiring the optimal generalization capability. Hence, the robustness in PGNN can be controlled by the criterion "How much computational complexity we can put up with?" under the constraint of Eq.(20).

## 6 Computer simulations

In this section, the proposed IPGNN is experimentally compared with the minimal resource allocating network (M-RAN) [33] and the back propagation (BP) algorithm [23]. In M-RAN, radial basis functions are adopted as basis functions, while BP employs sigmoidal activation functions.

Let us consider the following chaotic series created by the Mackey-Glass delay-difference equation:

$$g(t+1) = \begin{cases} (1-b)g(t) + a\dfrac{g(t-\tau)}{1+g(t-\tau)^{10}} \\ \qquad\qquad\qquad \text{for } t \geq \tau + 1, \\ 0.3 \qquad\qquad\quad \text{for } 1 \leq t \leq \tau, \end{cases} \tag{55}$$

where $a = 0.2$, $b = 0.1$, and $\tau = 17$. Let $\{h_t\}_{t=1}^{199}$ be

$$h_t = g(t + \tau + 1). \tag{56}$$

Our task is to estimate $\{h_t\}_{t=1}^{199}$ from 50 given sample values $\{y_i\}_{i=1}^{50}$:

$$y_i = h_p + n_i \ : \ p = \left\lceil 199 \times \frac{i}{50} \right\rceil, \tag{57}$$

where $\lceil c \rceil$ denotes the minimum integer larger than or equal to $c$ and $\{n_i\}_{i=1}^{50}$ are noises independently subject to the same normal distribution with mean 0 and variance 0.1:

$$n_i \sim N(0, 0.1). \tag{58}$$

Let us consider sample points $\{x_i\}_{i=1}^{50}$ corresponding to the sample values $\{y_i\}_{i=1}^{50}$:

$$x_i = -1 + \frac{1}{100} \times \left\lceil 199 \times \frac{i}{50} \right\rceil. \tag{59}$$

Then $f_0(-1 + \frac{t}{100})$ can be regarded as an estimate of $h_t$ for $1 \leq t \leq 199$, where $f_0(x)$ is a learning result function. The quality of the learning result function $f_0(x)$ is evaluated by the sum of the squared error:

$$\text{Error} = \sum_{t=1}^{199} \left( f_0(-1 + \frac{t}{100}) - h_t \right)^2. \tag{60}$$

Simulations are carried out in the following conditions.

**(a) IPGNN:** We adopt the polynomial space of order 19 as $H$, i.e., $H$ is spanned by $\{x^n\}_{n=0}^{19}$ and the inner product in $H$ is defined as

$$\langle f, g \rangle = \int_{-1}^{1} f(x)\overline{g(x)}dx. \tag{61}$$

Let $P_n(x)$ be the Legendre polynomial of order $n$ expressed as

$$P_n(x) = k_n x^n + k_n' x^{n-1} + \cdots. \tag{62}$$

Then the reproducing kernel of $H$ is given as follows [28]:

$$K(x,x') = \begin{cases} \dfrac{k_{19}}{k_{20}} \dfrac{P_{20}(x)P_{19}(x') - P_{19}(x)P_{20}(x')}{x - x'} \\ \qquad\qquad\qquad\qquad \text{if } x \neq x', \\ \dfrac{k_{19}}{k_{20}} \left(P'_{20}(x)P_{19}(x) - P'_{19}(x)P_{20}(x)\right) \\ \qquad\qquad\qquad\qquad \text{if } x = x'. \end{cases} \tag{63}$$

We construct the PGNN following Corollary 2. That is, for $m = 0, 1, \ldots, 19$, the hidden unit with the input-output function being $K(x, x_{m+1})$ is added, and the weights are determined by Eqs.(49) and (50). For $m = 20, 21, \ldots, 49$, weights are modified by Eq.(48).

**(b) M-RAN:** Parameters are assigned as $\epsilon_{max} = 0.18$, $\epsilon_{min} = 0.04$, $\gamma = 0.96$, $e_{min} = 0.004$, $e'_{min} = 0.006$, $\kappa = 0.8$, $P_0$=1, $P_n = 0.19$, $M = 30$, and $\delta = 0.000001$.

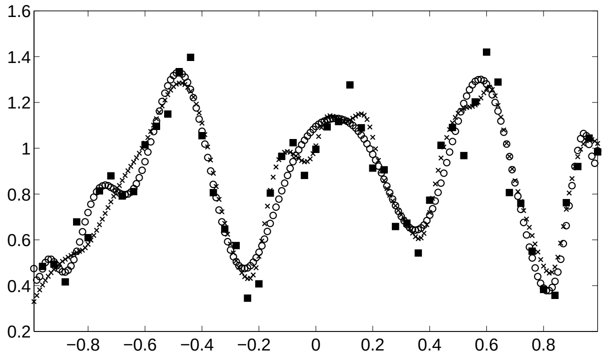
**(c) BP:** The number of hidden units is fixed to 20 throughout the learning process.

Note that IPGNN and M-RAN are incremental learning methods where each training example is used once and never used again, while BP is a batch learning method where all training examples are used again and again until convergence. In the cases of IPGNN and M-RAN, training examples are incrementally supplied from $(x_1, y_1)$ to $(x_{50}, y_{50})$.

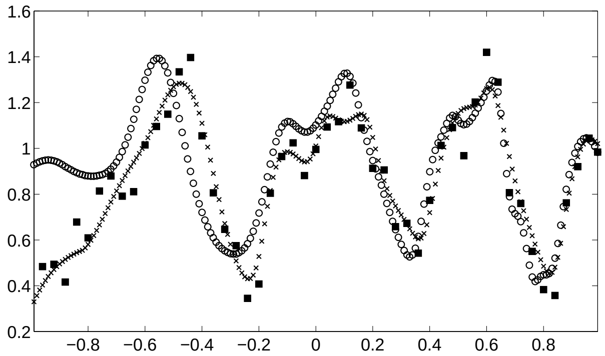
Estimation results of IPGNN and M-RAN are displayed in Fig.2 (a) and (b), respectively. '$\times$' and '$\circ$' denote the original series and estimation results, respectively. '▪' denotes training examples. The errors by IPGNN and M-RAN measured by Eq.(60) are 0.98 and 8.26, respectively. These results show that IPGNN provides better estimation than M-RAN. The estimation results of IPGNN are independent of the order of training examples, which is easily confirmed by the fact that IPGNN gives exactly the same learning results as those obtained by BPGNN. In contrast, old training examples tend to be forgotten in M-RAN since the existing parameters of the NN are adjusted to fit the additional training example if it has no novelty. Hence, M-RAN does not give good estimation around the region $[-1, -0.5]$ from which the old training examples were sampled.

Compared with M-RAN, IPGNN is quite easy to use in practice. In this simulation, the number of parameters which should be determined in IPGNN in advance is only one, the degree of the polynomial. Intuitively, this parameter determines the smoothness of the original series. This implies that the estimation result is not so sensitive to a small change in the parameter. Hence, it is not so difficult to tune the parameter in practice. On the other hand, M-RAN has many parameters which should be determined in advance, and estimation results are sensitive to small changes in any of the parameters.

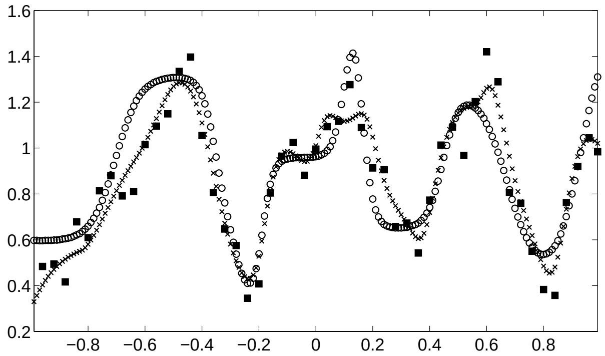
The estimation result of BP is shown in Fig.2 (c). The error by BP is 2.97. This result shows that IPGNN provides better estimation than BP. This may be explained by the fact that there are a lot of local minima in BP.



(a) IPGNN: Error = 0.98



(b) M-RAN: Error = 8.26



(c) BP: Error = 2.97

Figure 2: Estimation results of chaotic series created by the Mackey-Glass delay-difference equation from 50 training examples. '$\times$' and '$\circ$' denote the original series and estimation results, respectively. '■' denotes training examples.

## 7 Conclusion

We proposed an incremental learning method in neural networks under the projection learning criterion. The proposed method provides exactly the same learning result as that obtained by batch learning, and moreover it is more efficient in computation than batch learning. In our future work, we will give a robust construction method of NNs in an incremental manner.

In this paper, we focused on the case where the learning target function does not change during the learning process. However, it is practically very important to incorporate adaptability. Extending the proposed method to allow this challenging situation is extremely important and undoubtedly promising.

## Acknowledgement

The authors would like to thank anonymous reviewers for their valuable comments, which greatly improved the readability of this paper.

## A Proof of Theorem 1

It follows from Eq.(17) that

$$\begin{aligned} G_{N+1} &= \sum_{j=1}^{N+1} \left( e_j^{(N+1)} \otimes \overline{u_j} \right) \\ &= \Gamma_{N+1} G_N + e_{N+1}^{(N+1)} \otimes \overline{u_{N+1}}, \end{aligned} \tag{64}$$

where $\Gamma_{N+1}$ is defined by Eq.(25). When $u_{N+1} \notin \mathcal{L}\left(\{u_j\}_{j=1}^N\right)$, it follows from Theorem 4.3 in the reference [1] that Eq.(64) yields

$$\begin{aligned} G_{N+1}^\dagger &= G_N^\dagger \Gamma_{N+1}^* \\ &\quad + \tilde{u}_{N+1} \otimes \overline{(e_{N+1}^{(N+1)} - \Gamma_{N+1}(G_N^\dagger)^* u_{N+1})}. \end{aligned} \tag{65}$$

Let $Z_{N+1}$ be an arbitrary operator from $H$ to $\mathbf{C}^{N+1}$. It follows from Eqs.(18) and (65) that

$$\begin{aligned} W_{N+1} &= (G_{N+1}^\dagger)^* + (I_{N+1} - G_{N+1} G_{N+1}^\dagger) Z_{N+1} \\ &= \Gamma_{N+1}(G_N^\dagger)^* \\ &\quad + (e_{N+1}^{(N+1)} - \Gamma_{N+1}(G_N^\dagger)^* u_{N+1}) \otimes \overline{\tilde{u}_{N+1}} \\ &\quad + \Gamma_{N+1}(I_N - G_N G_N^\dagger)\Gamma_{N+1}^* Z_{N+1}. \end{aligned} \tag{66}$$

The prior result $W_N$ can be represented as

$$W_N = (G_N^\dagger)^* + (I_N - G_N G_N^\dagger) Z'_N, \tag{67}$$

where $Z'_N$ is an operator from $H$ to $\mathbf{C}^N$. Since $Z_{N+1}$ in Eq.(66) is arbitrary, it can be regarded as

$$Z_{N+1} = \Gamma_{N+1}\left(Z'_N(I_H - u_{N+1} \otimes \overline{\tilde{u}_{N+1}}) + Z_N\right), \tag{68}$$

where $Z_N$ is an arbitrary operator from $H$ to $\mathbf{C}^N$. Then Eqs.(66)–(68) yield

$$\begin{aligned} W_{N+1} &= \Gamma_{N+1}W_N + (e_{N+1}^{(N+1)} - \Gamma_{N+1}W_N u_{N+1}) \otimes \overline{\tilde{u}_{N+1}} \\ &\quad + \Gamma_{N+1}P_{\mathcal{N}(G_N^*)}Z_N. \end{aligned} \tag{69}$$

It follows from Proposition 2 that $\{u_j^*\}_{j=1}^{N+1}$ can be calculated as

$$u_j^* = W_{N+1}^* e_j^{(N+1)}. \tag{70}$$

Substituting Eq.(69) into Eq.(70) and letting $Y_N = Z_N^*$, we have Theorem 1. ■

## B Proof of Theorem 2

In this proof, let us denote the prior weight vector by $w_m^{(N)}$, and the posterior weight vector by $w_{m+1}^{(N+1)}$ or $w_{m+1}^{(N)}$. When $\psi_{m+1} \in \mathcal{R}(A_m^*)$, or $\psi_{m+1} \notin \mathcal{R}(A_m^*)$ and $\psi_{m+1} \in \mathcal{L}\left(\{u_j\}_{j=1}^N\right)$, Eqs.(20) and (21) yield

$$N \geq \operatorname{rank}(A_{m+1}^*) \tag{71}$$

$$\mathcal{L}\left(\{u_j\}_{j=1}^N\right) \supset \mathcal{R}(A_{m+1}^*). \tag{72}$$

Therefore, it follows from Eqs.(22) and (7) that weights are determined as

$$w_{m+1}^{(N)} = W_N f_{m+1}. \tag{73}$$

From Eqs.(73) and (33), we have Eq.(44) or (45). In the case where $\psi_{m+1} \notin \mathcal{R}(A_m^*)$ and $\psi_{m+1} \notin \mathcal{L}\left(\{u_j\}_{j=1}^N\right)$, if a novel hidden unit with the input-output function $u_{N+1}$ being a function in $\mathcal{R}(A_{m+1}^*)$ but not in $\mathcal{R}(A_m^*)$ is added to the prior NN, then it follows from Eqs.(20) and (21) that the posterior NN satisfies the following conditions:

$$N+1 \geq \operatorname{rank}(A_{m+1}^*), \tag{74}$$

$$\mathcal{L}\left(\{u_j\}_{i=1}^{N+1}\right) \supset \mathcal{R}(A_{m+1}^*). \tag{75}$$

Therefore, it follows from Eqs.(22) and (7) that weights are determined as

$$w_{m+1}^{(N+1)} = W_{N+1} f_{m+1}. \tag{76}$$

From Eqs.(76), (69), and (33), we have

$$\begin{aligned} w_{m+1}^{(N)} &= \Gamma_{N+1}W_N(f_m + \beta_{m+1}\zeta_{m+1}^{(b)}) \\ &\quad + \langle f_m + \beta_{m+1}\zeta_{m+1}^{(b)}, \tilde{u}_{N+1}\rangle \\ &\quad (e_{N+1}^{(N+1)} - \Gamma_{N+1}W_N u_{N+1}) + \Gamma_{N+1}z, \end{aligned} \tag{77}$$

where $z$ is an arbitrary vector in $\mathcal{N}(G_N^*)$. Since it follows from the reference [14] that

$$f_m \in \mathcal{R}(A_m^*), \tag{78}$$

Eq.(77) yields Eqs.(46) and (47). ■

## References

[1] A. Albert, Regression and the Moore-Penrose Pseudoinverse, Academic Press, New York and London, 1972.

[2] N. Aronszajn, “Theory of reproducing kernels,” Trans. Amer. Math. Soc., vol. 68, pp. 337–404, 1950.

[3] S. Amari, “Natural gradient works efficiently in learning,” Neural Computation, vol. 10, no. 2, pp. 251–276, 1998.

[4] R. J. Duffin and A. C. Schaeffer, “A class of non harmonic Fourier series,” Trans. on Amer. Math. Soc., vol. 72, pp. 341–366, 1952.

[5] F. Girosi, “An equivalence between sparse approximation and support vector machines,” Neural Computation, vol. 10, no. 6, pp. 1455–1480, 1998.

[6] B. A. Heil and D. Walnut, “Continuous and discrete wavelet transforms,” SIAM Rev., vol. 31 , pp. 628–666, 1989.

[7] H. Iwaki, H. Ogawa, and A. Hirabayashi, “Optimally generalizing neural networks with ability to recover from stuck-at $r$ faults,” IEICE Trans., vol. J83-D-II, no. 2, pp. 805–813, 2000. (In Japanese)

[8] V. Kadirkamanathan and M. Niranjan, “A function estimation approach to sequential learning with neural networks,” Neural Computation, vol. 5, no. 6, pp. 954–975, 1993.

[9] C. Molina and M. Niranjan, “Pruning with replacement on limited resource allocating networks by F-projections,” Neural Computation, vol. 8, no. 4, pp. 855–868, 1996.

[10] N. Murata, S. Yoshizawa and S. Amari, “Network information criterion — Determining the number of hidden units for an artificial neural network model,” IEEE Trans. Neural Networks, vol. 5, no. 6, pp. 865–872 (1994).

[11] A. Nakashima and H. Ogawa, “How to design a regularization term for improving generalization,” Proc. ICONIP’99, the 6th Intl. Conf. Neural Information Processing, vol. 1, pp. 222–227, Perth, Australia, 1999.

[12] S. Nakazawa and H. Ogawa, “Optimal realization of optimally generalizing neural networks,” IEICE Technical Report, NC96-60, pp. 17–24, 1996. (In Japanese)

[13] H. Ogawa, “A theory of pseudo biorthogonal bases,” IECE Technical Report, PRL77-60, pp. 69–78, 1978. (In Japanese)

[14] H. Ogawa, “Projection filter regularization of ill-conditioned problem,” Proc. SPIE, vol. 808, Inverse Problems in Optics, pp. 189–196, 1987.

[15] H. Ogawa, “Inverse problem and neural networks,” Proc. IEICE 2nd Karuizawa Workshop on Circuits and Systems, Karuizawa, Japan, pp. 262–268, 1989. (In Japanese)

[16] H. Ogawa, “Neural network learning, generalization and over-learning,” Proc. ICIIPS’92, Intl. Conf. Intelligent Information Processing & System, vol. 2, pp. 1–6, Beijing, China, 1992.

[17] H. Ogawa, “Theory of pseudo biorthogonal bases and its application,” Research Institute for Mathematical Science, RIMS Kokyuroku, No. 1067, Reproducing Kernels and their Applications, pp. 24–38, 1998.

[18] H. Ogawa and E. Oja, “Projection filter, Wiener filter, and Karhunen-Loève subspaces in digital image restoration,” J. Mathematical Analysis and Applications, vol. 114, no. 1, pp. 37–51, 1986.

[19] H. Ogawa, E. Oja and J. Lampinen, “Projection filters for image and signal restoration,” Proc. IEEE Intl. Conf. Systems Engineering, pp. 93–97, Dayton, USA, 1989.

[20] E. Oja and H. Ogawa, “Parametric projection filter for image and signal restoration,” IEEE Trans. Acoustics, Speech, and Signal Processing, vol. ASSP-34, no. 6, pp. 1643–1653, 1986.

[21] J. Platt, “A resource-allocating network for function interpolation,” Neural Computation, vol. 3, no. 2, pp. 213–225, 1991.

[22] R. Reed, “Pruning algorithms — A survey,” IEEE Trans. Neural Networks, vol. 4, no. 5, pp. 740–747, 1993.

[23] D. E. Rumelhart, G. E. Hinton and R. J. Williams, “Learning internal representations by error propagation,” Parallel Distributed Processing: Explorations in the Microstructure of Cognition, pp. 318–362, The MIT Press, Cambridge, MA, 1986.

[24] S. Schaal and C. G. Atkeson, “Constructive incremental learning from only local information,” Neural Computation, vol. 10, no. 8, pp. 2047–2084, 1998.

[25] R. Schatten, Norm Ideals of Completely Continuous Operators, Springer-Verlag, Berlin, 1970.

[26] M. Sugiyama and H. Ogawa, “Incremental projection learning for optimal generalization,” Neural Networks, vol. 14, no. 1, pp. 53–66, 2001.

[27] M. Sugiyama and H. Ogawa, “Properties of incremental projection learning,” Neural Networks, vol. 14, no. 1, pp. 67–78, 2001.

[28] G. Szegö, Orthogonal Polynomials, Amer. Math. Soc., Providence, Rhode Island, 1939.

[29] S. Vijayakumar and S. Schaal, “Local adaptive subspace regression,” Neural Processing Letters, vol. 7, no. 3, pp. 139–149, 1998.

[30] H. Wahba, Spline Model for Observational Data, Society for Industrial and Applied Mathematics, Philadelphia and Pennsylvania, 1990.

[31] Y. Yamashita and H. Ogawa, “Optimum image restoration and topological invariance,” IEICE Trans., vol. J75-D-II, no. 2, pp. 306–313, 1992.

[32] K. Yamauchi and N. Ishii, “An incremental learning method with recalling interfered patterns,” Proc. IEEE Intl. Conf. Neural Networks, vol. 6, pp. 3159–3164, 1995.

[33] L. Yingwei, N. Sundararajan and P. Saratchandran, “A sequential learning scheme for function approximation using minimal radial basis function neural networks,” Neural Computation, vol. 9, no. 2, pp. 461–478, 1997.