A Computational Study of Incremental Projection Learning in Neural Networks

HENDRI MURFI*, BENYAMIN KUSUMOPUTRO**
* Department of Mathematics, University of Indonesia, Depok 16424 – INDONESIA

** Faculty of Computer Science, University of Indonesia, Depok 16424 – INDONESIA

Abstract: One of the essences of supervised learning in neural network is generalization capability. It is an ability to give an accurate result for data that are not learned in learning process. One of supervised learning method that theoretically guarantees the optimal generalization capability is *projection learning*. The method was formulated as inverse problem from functional analytic point of view in reproducing kernel Hilbert space. This paper will describe a computational study to the incremental projection learning in neural networks, called *incremental projection generalizing neural networks*, for solving function approximation problem. The study is done based on computer simulations of a mathematics function as test problem. Aspects of study focus on model selection, generalization capability and number of neurons in hidden layer.

Key Words: supervised learning, incremental projection learning, generalization capability, artificial neural networks, function approximation problem

1. Introduction

One of the essences of supervised learning in neural network is generalization capability. It is an ability to give an accurate result for data that are not learned in learning process. The generalization capability is measured base on *generalization error*[1] value. One of supervised learning method that has goal to increase generalization capability directly is *projection learning* proposed by Ogawa [4]. Ogawa formulated the supervised learning method as inverse problem from functional analytic point of view. In this method, learning target and learning result are assumed to belong to a reproducing kernel Hilbert space. Then, variance of generalization error is minimized under the constraint of minimal value of bias. The minimal value of bias is gotten with orthogonal projection of learning target onto approximation space. This approach will guarantee theoretically to get optimal generalization capability.

In neural networks, it is often expected to further improve the generalization capability after the learning process has been completed. One of the common approaches is to add learning data to the neural networks. The learning method is generally called *incremental learning*. Projection learning has developed for incremental learning and known as *incremental projection learning* (IPL). Vijayakumar and Ogawa developed IPL with absence of noise [11]. Then, Sugiyama and Ogawa continued to develop for presence of noise [6][7]. Sugiyama and Ogawa also presented that the IPL provided exactly the same generalization capability as that obtained by *batch projection learning*. This result shows that generalization capability of incremental projection learning probably still can be increased when we work on an *active learning* [9].

This paper will describe a computational study to the IPL in neural networks, called *incremental projection generalizing neural networks* (IPGNN), for solving function approximation problem. The study was done based on computer simulations of a mathematics function as test problem. Aspects of study focused on model selection, generalization capability and number of neurons in hidden layer. We also compared generalization capability of the neural networks with other neural networks, they ware *incremental back propagation neural networks* (IBPNN) with *incremental gradient descent learning with momentum* [2][3] and *incremental radial basis function neural networks* (IRBFNN) with *incremental orthogonal least squares learning* [1][2].

[1] *Generalization error* is formulated as $E_n \|f_m - f\|^2$, E_n is expectation function; f is learning target function and f_m is learning result function from a set of m learning data. This formula expanded by Takemura be $\|E_n f_m - f\|^2 + E_n \|f_m - E_n f\|^2$. The first and second term is called the *bias* and *variance* of f_m.

The implementation used software Matlab at Pentium II machine with windows environment.

This paper is organized as follow: section 2 will describe architecture of neural networks and algorithm of IPGNN used in this paper. In section 3, computer simulation and a computational study based on the computer simulation will be presented. This paper will be closed with a summation in section 4.

2. Architecture and Algorithm

IPGNN is a three-layer feedforward neural network with architecture as describes in Fig. 1. The architecture has an input layer, a hidden layer and an output layer with only one neuron. Number of neuron in hidden layer grows as long as learning process. Learning starts with no neuron in hidden layer and grows by allocating a new neuron based on sampling function of the additional learning datum in used reproducing kernel Hilbert space. *Reproducing kernel*[2] functions are adopted as activation or basis functions in hidden layer. In this architecture, there is no weight on connection from input layer to hidden layer.

Once the learning process has terminated then approximation value for any input (x) is given by the formulation:

$$f_m(x) = \sum_{i=1}^{N} w_i u_i(x) \qquad (1)$$

Where x is an input vector, N is number of neuron in hidden layer, w_i is weight on connection from *i*th neuron in hidden layer to output layer and $u_i(x)$ is *i*th basis function on x.

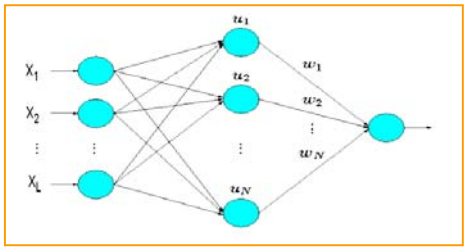

Fig. 1. Architecture of Neural Networks

[2] Let H be a functional Hilbert space and let D be the domain of the function in H. The *reproducing kernel* K(x,x') is bivariate function define on D x D that satisfies the following conditions:

- For any fixed x' in D, K(x,x') belongs to H as a function of x
- It hold that $\langle f(\cdot), K(\cdot, x')\rangle = f(x')$ for any f ∈ H, any x' ∈ D. Where $\langle \cdot,\cdot \rangle$ is an inner product defined on H

IPGNN is a neural networks that represents learning result function of IPL algorithm. Therefore, The neural networks learning problem is divided into two stages. Function approximation from given learning data is performed in the first stage, and a neural networks which represent the approximated function is constructed in the second stage. In this paper, we use IPL algorithm proposed by Sugiyama and Ogawa to develop function approximation [6][7]. The following proposition is a simpler form of IPL algorithm in the case where noise of learning data has normal distribution N(0, σ) and variance σ is positive.

Proposition 1 (IPL) [6][7] When the noise correlation matrix is positive definite and diagonal, a posterior projection learning result f_{m+1} is obtained by using prior results f_m as

$$f_{m+1} = \begin{cases} f_m + \dfrac{\beta'_{m+1} V_m'^{+} \psi_{m+1}}{v_2^{(m+1)}} & jika\ \psi_{m+1} \in R(A_m^*) \\ f_m + \dfrac{\beta'_{m+1} \tilde{\psi}_{m+1}}{v_1^{(m+1)}} & jika\ \psi_{m+1} \notin R(A_m^*) \end{cases} \qquad (2)$$

Where

$$V_m' = A_m^* Q_m^{-1} A_m \qquad (3)$$

$$\beta'_{m+1} = y_{m+1} - f_m(x_{m+1}) \qquad (4)$$

$$\tilde{\psi}_{m+1} = P_{N(A_m)} \psi_{m+1} \qquad (5)$$

$$v_1^{(m+1)} = \tilde{\psi}_{m+1}(x_{m+1}) \qquad (6)$$

$$v_2^{(m+1)} = \sigma_{m+1} + \left\langle V_m'^{+} \psi_{m+1}, \psi_{m+1} \right\rangle \qquad (7)$$

Where ψ_{m+1} is sampling function of m+1-th learning data, A_m is sampling operator, $R(A_m^*)$ is range of adjoin A_m or in this case is approximation space, $P_{N(A_m)}\psi$ is orthogonal projection of ψ onto null space of A_m, Q_m is noise correlation matrix and σ is noise variance.

Now we go to the second stage, the construction of a neural network that represents f_m. The construction algorithm is called *incremental projection learning in neural networks* (IPLNN). In this paper, we use IPLNN4 that constructs IPGNN which represents learning result function 2.2 [5][10] (Fig. 2).

In the IPLNN4 algorithm, the construction process is divided onto two categories base on sampling function of learning data (ψ_i). If $\psi_{m+1} \notin R(A_m^*)$ then a new neuron with the reproducing kernel $K(x,x_{m+1})$ as basis function is

added and weights on connection to output layer are adjusted. Otherwise, there is no additional neuron. There is only an adjusting of weights on connection to output layer. The condition $\psi_{m+1} \notin R(A_m^*)$ means that ψ_{m+1} is linearly independent of $\{\psi_j\}_{j=1}^m$, i.e., the approximation space $R(A_{m+1}^*)$ becomes wider than $R(A_m^*)$. In contrast, $\psi_{m+1} \in R(A_m^*)$ means that ψ_{m+1} is linearly dependent of $\{\psi_j\}_{j=1}^m$, and hence the approximation space $R(A_{m+1}^*)$ is equal to $R(A_m^*)$. The Condition $\psi_{m+1} \notin R(A_m^*)$ can be checked since $\psi_{m+1} \notin R(A_m^*)$ if and only if

$$P_{N(A_m)}\psi_{m+1} = \tilde{\psi}_{m+1} \neq 0 \qquad (8)$$

If dimension of reproducing kernel Hilbert space is μ then condition $N(A_\mu) = \{0\}$ will be reached when a learning data such that $\psi_{m+1} \notin R(A_m^*)$ is added μ times. Therefore, the number of neuron in hidden layer theoretically is less than or equal to the dimension of reproducing kernel Hilbert space.

In implementation, Sugiyama and Ogawa use the following criterion.

Jika $\|\tilde{\psi}_{m+1}\|^2 = v_1 > \varepsilon$ maka $\psi_{m+1} \notin R(A_m^*)$ (9)

Where $\in$ is small constant, say $\in = 10^{-4}$.

3. Computer Simulation

In this section, we shall describe simulation result of generalization capability of IPGNN. We use function 3.1 as test problem whose domain is [-Π,Π]. Generalization error of learning result is measured by equation 3.2.

$$f = 2x - 14e^{-3(x-2.5)^2} - 5e^{-6(x-0.5)^2} + 3e^{-3x^2} + 12e^{-(x+2.5)^2} \qquad (10)$$

$$Gen.err = \frac{1}{126}\sum_{n=0}^{125}[f(-\pi + 0.05n) - f_m(-\pi + 0.05n)]^2 \qquad (11)$$

Reproducing kernel Hilbert space is spanned by {1, sin kx, cos kx} whose inner product is defined as

$$\langle f, g\rangle = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)\overline{g(x)}dx \qquad (12)$$

and reproducing kernel is defined as

---

input $(x_{m+1}, y_{m+1}), \sigma_{m+1}$

if $m = 0$ {

generate first neuron in hidden layer

$u_1 \leftarrow K(x, x_1)$; $w_1 \leftarrow \frac{y_1}{\|\psi_1\|^2}$;

$C_1^{(1)} \leftarrow \frac{\sigma_1}{\|\psi_1\|^4}$; $D_1^{(1)} \leftarrow \frac{1}{\|\psi_1\|^2}$;

$N \leftarrow 1$;

} *else* {

$[b_N^{(m+1)}]_i \leftarrow \langle u_i, \psi_{m+1}\rangle$;

$[c_N^{(m+1)}]_i \leftarrow \sum_{j=1}^{N}[C_N^{(m)}]_{ij}\overline{[b_N^{(m+1)}]_j}$;

$[d_N^{(m+1)}]_i \leftarrow \sum_{j=1}^{N}[D_N^{(m)}]_{ij}\overline{[b_N^{(m+1)}]_j}$;

$\beta_{m+1} \leftarrow y_{m+1} - \sum_{i=1}^{N} w_i[b_N^{(m+1)}]_i$;

$v_1^{(m+1)} \leftarrow \psi_{m+1}(x_{m+1}) - \sum_{i=1}^{N}[c_N^{(m+1)}]_i[b_N^{(m+1)}]_i$;

$v_2^{(m+1)} \leftarrow \sigma_{m+1} + \sum_{i=1}^{N}[d_N^{(m+1)}]_i[b_N^{(m+1)}]_i$;

if $v_1^{(m+1)} \neq 0$ {

add the $(N+1)-st$ *neuron in hidden layer*

$u_{N+1} \leftarrow K(x, x_{m+1})$, $w_{N+1} \leftarrow \frac{\beta_{m+1}}{v_1^{(m+1)}}$;

$w^{(N)} \leftarrow w^{(N)} - \frac{\beta_{m+1}}{v_1^{(m+1)}}c_N^{(m+1)}$;

$C_{N+1}^{(m+1)} \leftarrow \Gamma_{N+1}C_N^{(m)}\Gamma_{N+1}^* + \frac{a_{N+1}^{(m+1)} \otimes \overline{a_{N+1}^{(m+1)}}}{v_1^{(m+1)}}$

$D_{N+1}^{(m+1)} \leftarrow \Gamma_{N+1}C_N^{(m)}\Gamma_{N+1}^* + \frac{v_2^{(m+1)}(a_{N+1}^{(m+1)} \otimes \overline{a_{N+1}^{(m+1)}})}{v_1^{(m+1)^2}}$

$- \frac{\Gamma_{N+1}d_N^{(m+1)} \otimes \overline{a_{N+1}^{(m+1)}} + a_{N+1}^{(m+1)} \otimes \overline{\Gamma_{N+1}d_N^{(m+1)}}}{v_1^{(m+1)}}$;

$N \leftarrow N+1$;

} *else* {

$w^{(N)} \leftarrow w^{(N)} + \frac{\beta_{m+1}d_N^{(m+1)}}{v_2^{(m+1)}}$;

$C_N^{(m+1)} \leftarrow C_N^{(m)}$;

$D_N^{(m+1)} \leftarrow D_N^{(m)} - \frac{d_N^{(m+1)} \otimes \overline{d_N^{(m+1)}}}{v_2^{(m+1)}}$;

}

}

Fig. 2. The IPLNN4 algorithm. [ ]$_i$ and [ ]$_{ij}$ denote the i-th element of a vector and the (i,j)-element of a matrix, respectively.

$$K(x,x') = \begin{cases} 2k+1 & jika\ x = x' \\ \dfrac{\sin\dfrac{(2k+1)(x-x')}{2}}{\sin\dfrac{x-x'}{2}} & jika\ x \neq x' \end{cases} \quad (13)$$

Our simulation starts with simulation of optimal order basis selection based on number and noise variance of learning data. In learning point of view, the selection method is known as *model selection*. From Table 1 and Table 2, we can see that optimal order basis depend on both number and noise variance of learning data. Our result says that we need larger order basis for larger number or smaller noise variance of learning data. Base on this situation, we also need incremental model selection in order to get optimal generalization capability of incremental projection learning. The model selection method that heavily related to incremental projection learning has been proposed but the method has not aimed at incremental learning scenario yet [8].

From Table 1 and Table 2, we also can see that number of hidden neuron is bigger than dimension of used Hilbert space. The number of hidden neuron does not depend on dimension of Hilbert space but have tendency to depend on number of learning data.

Table 1. Simulation of optimal order basis selection based on umber of learning data

Case	Number of Data	Noise	Order Basis	Dim of Space	Number of Neuron
I	40	N(0,1)	5	11	40
	30	N(0,1)	4	9	30
	20	N(0,1)	4	9	20
	10	N(0,1)	4	9	10
	5	N(0,1)	4	9	5
II	40	N(0,1)	5	11	40
	30	N(0,1)	4	9	30
	20	N(0,1)	4	9	20
	10	N(0,1)	4	9	10
	5	N(0,1)	4	9	5
III	40	N(0,0.1)	6	13	40
	30	N(0,0.1)	4	9	30
	20	N(0,0.1)	6	13	20
	10	N(0,0.1)	4	9	10
	5	N(0,0.1)	4	9	5

Table 2. Simulation of optimal order basis selection based on noise variance of learning data

Case	Number of Data	Noise	Order Basis	Dim of Space	Number of Neuron
I	40	N(0,0.1)	6	13	40
	40	N(0,0.1)	6	13	40
	40	N(0,0.1)	6	13	40
II	40	N(0,0.2)	5	11	40
	40	N(0,0.2)	6	13	40
	40	N(0,0.2)	6	13	40
III	40	N(0,0.3)	6	13	40
	40	N(0,0.3)	6	13	40
	40	N(0,0.3)	6	13	40
IV	40	N(0,2)	5	11	40
	40	N(0,2)	4	9	40
	40	N(0,2)	5	11	40
V	40	N(0,3)	4	9	40
	40	N(0,3)	4	9	40
	40	N(0,3)	4	9	40
VI	40	N(0,4)	4	9	40
	40	N(0,4)	4	9	40
	40	N(0,4)	4	9	40

Next, we will compare generalization capability of IPGNN with other neural networks; they are IBPNN and IRBFNN. Characteristic of each neural network as following

- IBPNN. The optimal number of hidden units is choose manually based on characteristic of learning data and fixed throughout the learning process. Tag-sigmoid function is adopted as activation function in hidden layer and linear function in output layer. The stopping criteria is if epoch is bigger than 3000 or mean square error is smaller than 10^{-5} on minimum generalization error has reached
- IRBFNN. The number of hidden units is dynamically. Learning starts with no neuron in hidden layer and grows by allocating a new neuron in hidden layer. Radial basis function is adopted as activation function in hidden layer and linear function in output layer. The stopping criteria is if the maximum neuron in hidden layer has reached or sum square error is smaller than 10^{-5}.

The computer simulation is grouped based on number and noise variance of learning data. Our simulation starts with forty learning data whose normal distribution with variance 1. Table 3 is simulation result for this condition. From the table, we can see that IPGNN does not always give better

generalization capability. The results depend heavily on the learning data.

Table 3. Generalization capability
for forty learning data and noise variance 1

Case	Number of Data	Noise	IPGNN	IBPNN	IRBFNN
I	40	N(0,1)	0.3999	0.8573	1.1943
II	40	N(0,1)	1.1037	0.8904	3.5178
III	40	N(0,1)	0.9641	0.6966	0.7221

Next, we will simulate generalization capability for some kind of number of learning data. Table 4 describes the condition. From the table, we can see that IPGNN still gives better generalization capability when number of learning data is small enough. In case II and III, we also can see that IPGNN gives better generalization capability although when number of learning data is big enough it gives worse generalization capability. Graphical simulation of case III with five learning data is presented by Fig. 3.

Table 4. Generalization capability
for some kind of number of learning data

Case	Number of Data	Noise	IPGNN GE	IBPNN GE	IRBFNN GE
I	40	N(0,1)	0.3999	0.7542	1.1943
	30	N(0,1)	0.8083	0.7526	0.4486
	20	N(0,1)	1.2895	0.7235	0.6550
	10	N(0,1)	0.5823	4.3613	0.7283
	5	N(0,1)	1.3314	6.4306	1.8085
II	40	N(0,1)	0.9641	0.6758	0.7596
	30	N(0,1)	1.0334	0.7864	0.7318
	20	N(0,1)	2.5886	0.8808	3.1920
	10	N(0,1)	0.9919	3.7826	1.1394
	5	N(0,1)	1.6796	7.6127	2.1461
III	40	N(0,0.1)	0.6017	0.2202	0.4931
	30	N(0,0.1)	1.8677	0.3725	2.0004
	20	N(0,0.1)	1.9294	0.3560	1.9839
	10	N(0,0.1)	0.4918	3.9742	0.4646
	5	N(0,0.1)	1.2998	8.9038	2.4679

The next simulation, we shall simulate generalization capability based on noise variance of learning data. We use big enough noise variance of learning data (bigger than one). Table 5 presents simulation result of this condition. From this table, we can find out that IPGNN still gives better generalization capability when noise variance of learning data is big enough. The Fig. 4 simulates case V of Table 5 graphically.

Finally, we shall simulate the condition of learning data whose number is big enough and noise variance is small enough. For this condition, we get result that IPGNN gives worse generalization capability than IBPNN. While than IRBFNN, IPGNN can give either better or worse result. Table 6 and Fig. 5 present simulation of this condition.

Table 5. Generalization capability
for big enough noise variance of learning data

Case	Number of Data	Noise	IPGNN	IBPNN	IRBFNN
I	40	N(0,2)	1.3102	3.2028	4.6839
II	40	N(0,2)	1.6034	2.5719	11.5226
III	40	N(0,2)	0.8587	2.5123	1.5664
IV	40	N(0,3)	4.1020	7.1684	18.9112
V	40	N(0,3)	1.6403	4.5324	2.6182
VI	40	N(0,3)	2.5363	4.7056	8.5409
VII	40	N(0,4)	2.2749	8.1463	4.5663
VIII	40	N(0,4)	5.6786	7.9739	16.1306
IX	40	N(0,4)	5.0104	8.6923	8.5244

Table 6. Generalization capability for big enough number and small enough noise variance of learning data

Case	Number of Data	Noise	IPGNN	IBPNN	IRBFNN
I	40	N(0,0.1)	0.5265	0.3622	0.3673
II	40	N(0,0.1)	0.5718	0.2785	0.4741
III	40	N(0,0.1)	0.6082	0.3645	0.4056
IV	40	N(0,0.2)	0.6706	0.2388	0.4507
V	40	N(0,0.2)	0.6250	0.3099	0.4215
VI	40	N(0,0.2)	0.5761	0.2416	0.3307
VII	40	N(0,0.3)	0.7321	0.2314	1.0656
VIII	40	N(0,0.3)	0.5102	0.3278	0.5577
IX	40	N(0,0.3)	0.6315	0.2604	1.4034

4. Summary and Concluding Remark

In this paper we study computationally to incremental projection generalizing neural networks (IPGNN) for solving function approximation problem. Base on our simulation results, we have the following condition:

- The optimal model of IPGNN, in this manner is order basis, depends on both number of learning data and noise variance of learning data. We need larger order basis for larger number of learning data or smaller noise variance of learning data. Base on this situation, we also need incremental model selection in order to get optimal generalization capability of IPGNN. The model selection method that heavily related to incremental projection learning has been proposed but the method has not aimed at incremental learning scenario yet.

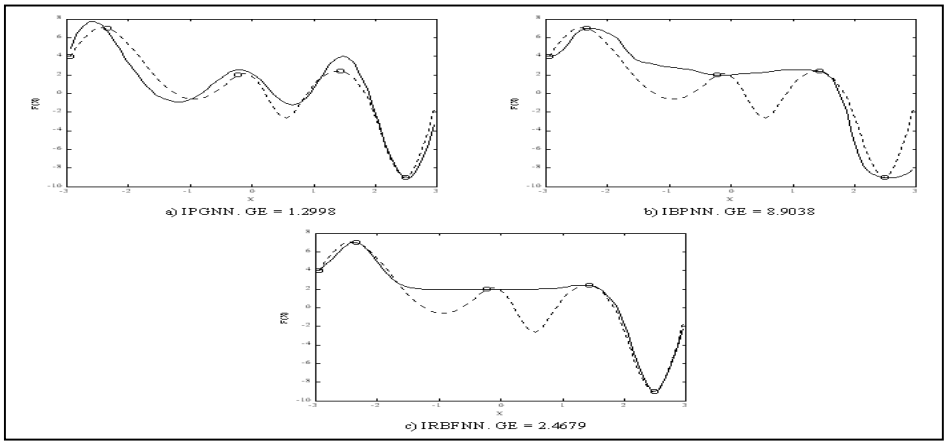

Fig. 3 Graphical simulations for case III with 5 learning data from Table 4. Solid and dotted lines denote the original function and a learning result. o indicates learning data.

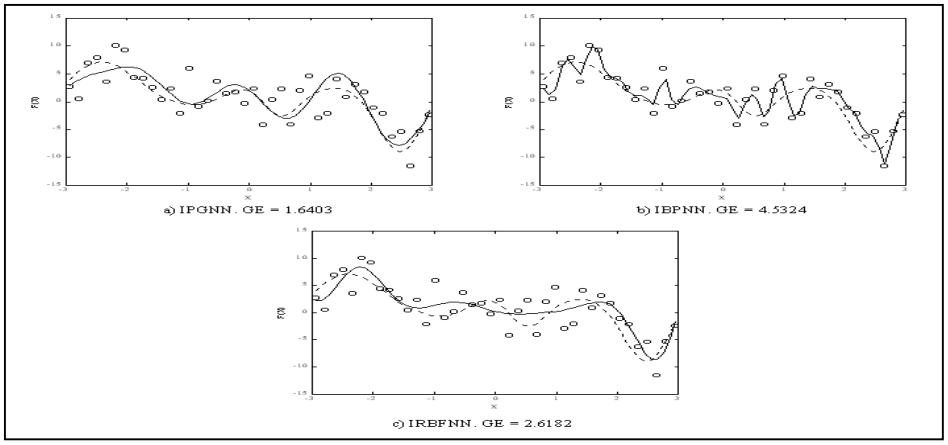

Fig. 4. Graphical simulations for case V from Table 5. Solid and dotted lines denote the original function and a learning result. o indicates learning data.

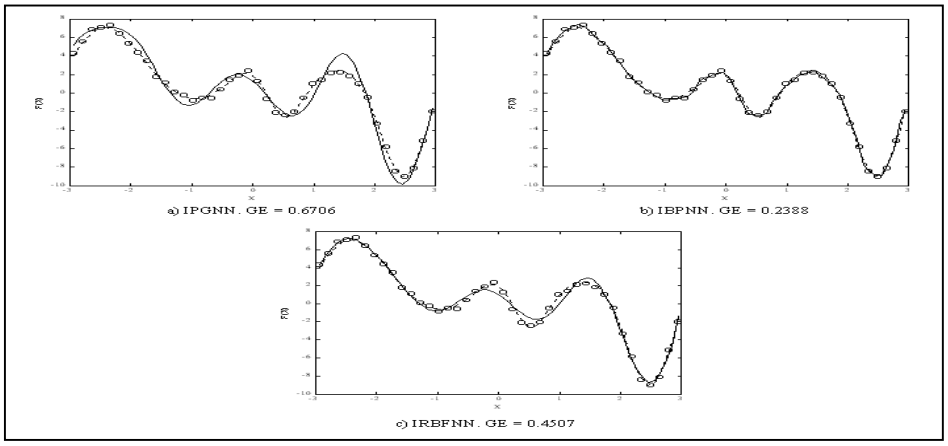

Fig. 5 Graphical simulations for case IV from Table 6. Solid and dotted lines denote the original function and a learning result. o indicates learning data.

- The number of neuron in hidden layer of IPGNN theoretically should be less than or equal to dimension of used reproducing kernel Hilbert space. In our simulation, the number of hidden neuron is bigger than dimension of the used reproducing kernel Hilbert space. The number of hidden neuron does not depend on dimension of Hilbert space but have tendency to depend on number of learning data.
- IPGNN does not always give better generalization capability. It still always gives better generalization capability when number of learning data is small enough or noise variance of learning data is big enough. Otherwise, IPGNN does not always give better generalization capability. Even though, when number of learning data is big enough and noise variance of learning data is small enough, IPGNN gives worse generalization capability than IBPNN

Reference

[1] Chen, S., C.F.N. Cowan and P. M. Grant. *Orthogonal Least Squares Learning Algorithm for Radial Basis Functions Networks*. IEEE Transaction on Neural Networks, Vol. 2, No. 2, pp. 302-309, 1991

[2] Demuth, H. and M. Beale. *Neural Network Toolbox 4.0 User's Guide for Matlab*. The MathWorks Inc., 2000

[3] M.T. Hagan, H.B. Demuth, M.H. Beale. *Neural Networks Design*. PWS Publishing Company, Boston, MA, 1996

[4] Ogawa, H. *Neural Networks Learning, Generalization and Over Learning*. Proceedings of the ICIIPS'92, International Conference on Intelligent Information Processing System, Beijing, China, 1992

[5] Sugiyama, M. and H. Ogawa. *Exact Incremental Projection Learning in Neural Networks*. IEICE Technical Report, NC98-97, pp. 149-156, 1999

[6] Sugiyama, M. and H. Ogawa. *Incremental Projection Learning for Optimal Generalization*. Neural Networks, Vol. 14, no. 1, pp. 53-66, 2001

[7] Sugiyama, M. and H. Ogawa. *Properties of Incremental projection learning*. Neural Networks, Vol. 14, no. 1, pp. 53-66, 2001

[8] Sugiyama, M. and H. Ogawa. *Subspace Information Criterion for Model Selection*. Neural Computation, vol.13, no.8, pp.1863-1889, 2001

[9] Sugiyama, M and H. Ogawa. *Active learning for optimal generalization in trigonometric polynomial model*. To appear in IEICE Transactions on Fundamental of Electronic, Communication and Computer Science

[10] Sugiyama, M and H. Ogawa. *Incremental Projection Learning for Optimal Generalization in Neural Networks.* (Submitted). Also as Technical Report, Departmen of Computer Science, Tokyo Institute of Technology, Japan

[11] Vijayakumar, S. and H. Ogawa. *RHKS Based Functional Analysis for Exact Incremental Learning, Neurocomputing (special issue on theoretical analysis of real valued function classes)*. Amsterdam: Elsevier Science, 29 (1-3), 85-113, 1998