A Two-Level Learning Hierarchy for Constructing Incremental Projection Generalizing Neural Networks and Its Application in Artificial Odor Discrimination System

HENDRI MURFI[1] and BENYAMIN KUSUMOPUTRO[2]
[1]Department of Mathematics
[2]Faculty of Computer Science
University of Indonesia - Depok 16424
INDONESIA

Abstract: - One of incremental learning-based neural networks that theoretically guarantees the optimal generalization capability and provides exactly the same generalization capability as that obtained by batch learning is incremental projection generalizing neural networks. This paper will describe a two-level learning hierarchy for constructing the networks. An incremental projection learning in neural networks algorithm is employed at the lower level to construct the network while the learning parameters, the orders of the reproducing kernel Hilbert space, are optimized using a genetic algorithm at the upper level. The networks produced by this learning hierarchy will be used as subsystem of the artificial odor discrimination system to approximate percentage of alcohol.

Key-Words: - incremental learning, incremental projection generalizing neural networks, genetic algorithm, artificial odor discrimination system

1 Introduction

One of the essences of supervised learning in neural network is generalization capability. This is an ability to give an accurate result for data that are not learned in learning process. 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*. Many incremental learning-based neural networks have been devised so far, such as resource allocating networks (RAN) proposed by Platt [7], interpretation of RAN from functional analytic point of view proposed by Kadirkamanathan et al [3] and minimal resource allocating networks (M-RAN) proposed by Yingwei et al [12]. Although RAN and its derivatives improve the efficiency of computation, the optimal generalization capability is not theoretically guaranteed and they provide poor generalization capability than batch learning. One of incremental learning-based neural networks that theoretically guarantees the optimal generalization capability and provides exactly the same learning result as that obtained by batch learning even in the non-asymptotic case is *incremental projection generalizing neural networks* (IPGNN) proposed by Sugiyama and Ogawa [8][10]. The network is trained with *incremental projection learning in neural networks* (IPLNN) algorithm. The algorithm is formulated from functional analytic point of view in a reproducing kernel Hilbert space. When the orders of the space are fixed, the network has a linear-in-the-parameters structure. It means that learning algorithm of the network is a linear learning problem.

In this paper, we will describe the application of IPGNN. Due to the generalization capability of IPGNN is a complex multimodal function on the space of the orders of the reproducing kernel Hilbert space [5], we adopt a two-level learning hierarchy for constructing IPGNN [1]. The learning strategy is based on the combined genetic algorithm (GA) and IPLNN algorithm. The orders of the reproducing kernel Hilbert space are optimized using the GA at the upper level. Given these parameters, the IPLNN algorithm is used to construct IPGNN at the lower level. The network produced by this learning hierarchy will be used as subsystem of the artificial odor discrimination (AOD) system to approximate percentage of alcohol.

2 Architecture and Algorithm

IPGNN is a three-layer feed forward neural network whose architecture describes in Fig. 1. The architecture has an input layer, a hidden layer and an output layer with only one neuron. There is no weight on connection from input layer to hidden layer. *Reproducing kernel* functions are adopted as activation functions in hidden layer. Number of neurons in hidden layer grows as long as learning process. Network starts with no neuron and grows by allocating a new neuron based on sampling function of the learning data on the approximation space that represented with hidden layer. If the sampling function is on the approximation space, or the reproducing kernel function on the new learning data is linearly independent of the activation functions of hidden neuron, then a new neuron is added and the reproducing kernel function on the new learning data is set to as activation function. Next, 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 weights are updated using incremental projection learning (IPL) criterion.

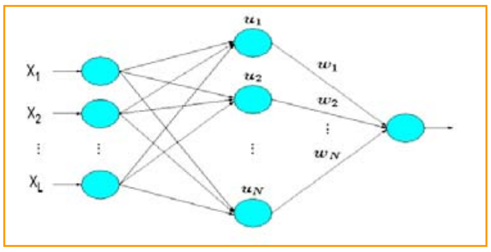

Fig. 1. Architecture of IPGNN

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

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

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.

IPL is incremental version of projection learning proposed by Ogawa [6]. Ogawa formulated the learning method as inverse problem from functional analytic point of view. In this method, learning target function and learning result function are assumed to belong to a reproducing kernel Hilbert space. Next, the variance of learning result function is minimized under the constraint of reducing the bias of learning result function to a certain level. This is done with orthogonal projection technique. This approach will guarantee theoretically for obtaining optimal generalization capability.

Definition 1 [6] An operator X_m is called the projection-learning operator if X_m minimizes the functional

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

under the constraint

$$X_m A_m = P_{R(A_m^*)} \tag{3}$$

where E_n is expectation function, n is noise of m learning data, A^*_m is adjoint of sampling operator A_m, $R(A^*_m)$ is range of A^*_m or approximation space in this case, P is orthogonal projection operator.

IPL has developed for some conditions. Vijayakumar and Ogawa developed IPL with absence of noise [11]. Then, Sugiyama and Ogawa continued to develop for presence of noise [9]. In this paper we will use IPL proposed by Sugiyama and Ogawa. 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 [9] 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

$$f_{m+1} = \begin{cases} f_m + \dfrac{\beta'_{m+1} V'^{+}_m \psi_{m+1}}{v_2^{(m+1)}} & \text{if } \psi_{m+1} \in R(A_m^*) \\ f_m + \dfrac{\beta'_{m+1} \tilde{\psi}_{m+1}}{v_1^{(m+1)}} & \text{if } \psi_{m+1} \notin R(A_m^*) \end{cases} \tag{4}$$

where

$$V'_m = A_m^* Q_m^{-1} A_m \tag{5}$$

$$\beta'_{m+1} = y_{m+1} - f_m(x_{m+1}) \tag{6}$$

$$\tilde{\psi}_{m+1} = P_{N(A_m)} \psi_{m+1} \tag{7}$$

$$v_1^{(m+1)} = \tilde{\psi}_{m+1}(x_{m+1}) \tag{8}$$

$$v_2^{(m+1)} = \sigma_{m+1} + \left\langle V'^{+}_m \psi_{m+1}, \psi_{m+1} \right\rangle \tag{9}$$

where ψ_{m+1} is sampling function of m+1-th learning data, $N(A_m)$ is null space of A_m, Q_m is noise correlation matrix and σ is noise variance.

In development, the neural networks learning problem is divided into two stages. Function approximation of given learning data is performed in the first stage (IPL algorithm), and a neural networks which represent the approximated function is constructed in the second stage. The constructing algorithm is called *incremental projection learning in neural networks* (IPLNN) algorithm. There are some proposed IPLNN, in this paper we use the efficient one called IPLNN4 algorithm. The algorithm constructs IPGNN that represents learning result function as describes in Eq. (4) [8][10] (Fig. 2).

```
$input(x_{m+1}, y_{m+1}), \sigma_{m+1}$
$if\ m=0\{$
    $generate\ first\ neuron\ in\ hidden\ layer$
    $u_1 \leftarrow K(x,x_1);\quad w_1 \leftarrow \frac{y_1}{\|\psi_1\|^2};$
    $C_1^{(1)} \leftarrow \frac{\sigma_1}{\|\psi_1\|^4};\quad D_1^{(1)} \leftarrow \frac{1}{\|\psi_1\|^2};$
    $N \leftarrow 1;$
$\}\ else\{$
    $[b_N^{(m+1)}]_i \leftarrow \langle u_i, \psi_{m+1}\rangle;\ [c_N^{(m+1)}]_i \leftarrow \sum_{j=1}^{N} [C_N^{(m)}]_{ij} \overline{[b_N^{(m+1)}]_j};$
    $[d_N^{(m+1)}]_i \leftarrow \sum_{j=1}^{N} [D_N^{(m)}]_{ij} \overline{[b_N^{(m+1)}]_j};\ \beta_{m+1} \leftarrow y_{m+1} - \sum_{i=1}^{N} w_i [b_N^{(m+1)}]_i;$
    $v_1^{(m+1)} \leftarrow \psi_{m+1}(x_{m+1}) - \sum_{i=1}^{N} [c_N^{(m+1)}]_i [b_N^{(m+1)}]_i;$
    $v_2^{(m+1)} \leftarrow \sigma_{m+1} + \sum_{i=1}^{N} [d_N^{(m+1)}]_i [b_N^{(m+1)}]_i;$
    $if\ v_1^{(m+1)} \neq 0\{$
        $add\ the\ (N+1)-st\ neuron\ in\ hidden\ layer$
        $u_{N+1} \leftarrow K(x, x_{m+1}),\ w_{N+1} \leftarrow \frac{\beta_{m+1}}{v_1^{(m+1)}};$
        $w^{(N)} \leftarrow w^{(N)} - \frac{\beta_{m+1}}{v_1^{(m+1)}} c_N^{(m+1)};$
        $C_{N+1}^{(m+1)} \leftarrow \Gamma_{N+1} C_N^{(m)} \Gamma_{N+1}^* + \frac{a_{N+1}^{(m+1)} \otimes \overline{a_{N+1}^{(m+1)}}}{v_1^{(m+1)}}$
        $D_{N+1}^{(m+1)} \leftarrow \Gamma_{N+1} C_N^{(m)} \Gamma_{N+1}^* + \frac{v_2^{(m+1)} (a_{N+1}^{(m+1)} \otimes \overline{a_{N+1}^{(m+1)}})}{v_1^{(m+1)^2}}$
            $- \frac{\Gamma_{N+1} d_N^{(m+1)} \otimes \overline{a_{N+1}^{(m+1)}} + a_{N+1}^{(m+1)} \otimes \overline{\Gamma_{N+1} d_N^{(m+1)}}}{v_1^{(m+1)}};$
        $N \leftarrow N+1;$
    $\}\ else\{$
        $w^{(N)} \leftarrow w^{(N)} + \frac{\beta_{m+1} d_N^{(m+1)}}{v_2^{(m+1)}};\ C_N^{(m+1)} \leftarrow C_N^{(m)};$
        $D_N^{(m+1)} \leftarrow D_N^{(m)} - \frac{d_N^{(m+1)} \otimes \overline{d_N^{(m+1)}}}{v_2^{(m+1)}};$
    $\}$
$\}$
```

Fig. 2. IPLNN4 Algorithm

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

To check $\psi_{m+1} \notin R(A_m^*)$, Sugiyama and Ogawa used the condition that $\psi_{m+1} \notin R(A_m^*)$ if and only if

$$P_{N(A_m)} \psi_{m+1} = \tilde{\psi}_{m+1} \neq 0 \tag{10}$$

In implementation, we use the following criterion.

$$\text{if } \|\tilde{\psi}_{m+1}\|^2 = v_1 > \varepsilon \text{ then } \psi_{m+1} \notin R(A_m^*) \tag{11}$$

where ε is a small constant, say $\varepsilon = 10^{-4}$.

3 The Combined GA and IPLNN

IPLNN algorithm is developed in a reproducing kernel Hilbert space. The quality of the learning result of IPLNN depends heavily on the choice of the spaces. After we choose a specific reproducing kernel Hilbert space, then the quality will depend on the choice of orders of the space.

In this paper, we will use trigonometric polynomial space as reproducing kernel Hilbert space. The space is spanned by $\{1, \sin kx, \cos kx\}_{k=1}^N$ defined on [-π, π], where N is order of the space. Reproducing kernel function in the space is defined as

$$K(x,x') = \begin{cases} 2N+1 & if\ x = x' \\ \dfrac{\sin\dfrac{(2N+1)(x-x')}{2}}{\sin\dfrac{x-x'}{2}} & if\ x \neq x' \end{cases} \tag{12}$$

If the dimension of input vector x is one as described above, then the space is called one-dimensional trigonometric polynomial space. The others are called multi-dimensional trigonometric polynomial space. Those are spanned by $\{1, \sin n_l \xi^{(l)}, \cos n_l \xi^{(l)}\}_{n_l=1}^{N_l}$, where l = 1, 2, .. L, L is dimension of input vector and N_l is order of space of l-*th* dimension. Reproducing kernel function is defined as

$$K(x,x') = \prod_{l=1}^{L} K_l(\xi^{(l)},\xi^{(l)'}) \tag{13}$$

where

$$K(\xi^{(l)},\xi^{(l)'}) = \begin{cases} \dfrac{\sin(2N_l+1)(\xi^{(l)}-\xi^{(l)'})/2}{\sin(\xi^{(l)}-\xi^{(l)'})/2} & if\, \xi^{(l)} \neq \xi^{(l)'} \\ 2N_l+1 & if\, \xi^{(l)} = \xi^{(l)'} \end{cases} \tag{14}$$

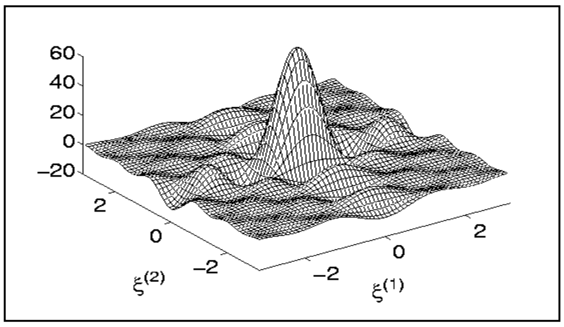

Fig. 3. Profile of reproducing kernel function K(x, x') of two-dimensional trigonometric polynomial space of order (3, 5) with x' = $(0, 0)^T$.

Due to the generalization capability of IPGNN is a complex function on the space of the orders of the reproducing kernel Hilbert space, we adopt a two-level learning schema proposed by Chen. The learning combines the GA and IPLNN algorithm, as illustrated in Fig. 4. At the upper level, the GA, with a population size of *p*, learns the orders of reproducing kernel Hilbert space N_1, N_2, … N_n based on the fitness function values provided by the lower level, where n is dimension of input vector. The lower level consists of the p parallel IPLNN algorithms, one for each of N_{1i}, N_{2i}, … N_{ni} provided by GA. In simulation, the data set is divided into a training set and validating set. The *i*th IPLNN algorithm constructs an IPGNN using the training data set with given N_{1i}, N_{2i}, … N_{ni}. The generalization capability, the mean square error (MSE) over the validation data set, of the resulting IPGNN is computed. The inverse of this generalization capability is the fitness function value f_i for the given N_{1i}, N_{2i}, … N_{ni}.

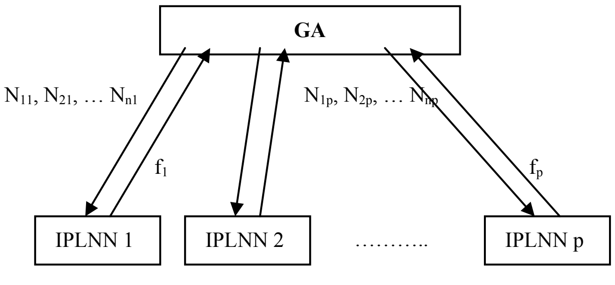

Fig. 4. Schematic of two-level learning hierarchy for IPGNN

The GA searches the solution space of a function through the use of simulated evolution, i.e., the survival of the fittest strategy. In general, the fittest individuals of any population tend to reproduce and survive to the next generation, thus improving successive generations. The GA explores all regions of the state space and exponentially exploiting promising areas through mutation, crossover, and selection operations applied to individuals in the population. First, The population is initialized randomly while the best individual found up to that point is copied into the newly generated population. This is repeated until no further improvement is evidenced. In our implementation, we use a modified GA toolbox proposed by Houck [2]. The orders of reproducing kernel Hilbert space, N_1, N_2, … N_n, are each coded into 16-bit string, and a population size is set to 5. We use simple crossover and binary mutation. Roulette selection is employed to determine parents for reproduction.

4 Computer Simulations

In this section, we describe computer simulation of generalization capability of IPGNN as subsystem of AOD system to approximate percentage of alcohol.

Generally, The AOD system consists of three subsystems. They are sensor system, electronic system (frequency counter) and neural networks system. Some quartz sensors with a chemical membrane construct the sensor system. In order that an odor can be recognized by the system, we must heat the odor first. Next, the odor will result molecules that will be absorbed by membrane of sensor of sensor system. This will be decreased basic frequency of those sensors as much as weight of molecules of the odor. The change of frequency will be converted by electronic system (frequency counter) into numbers that represent the odor. Finally, Neural network system is used to recognize the odor based on pattern of those numbers.

The goal of our simulation in this section is to know how capable IPGNN to support AOD system in approximating percentage of alcohol for small-number training data. To do this, we will use the data of alcohol resulted by the four-sensor AOD system [4]. Those are data of percentage of alcohol that are in interval 0% up to 35%. The data are only consisted of four groups. They are alcohol 0%, alcohol 15%, alcohol 25% and alcohol 35%. Next, The data is divided into a training set and a validation set. The validation data is set to 220 data while the training data is set to some number of data. We start our simulation from small-number training data and increase until we have good-enough generalization capability. Each simulation is consisted four categories based on which group of data that is not trained to network. Moreover, the data is called as *unknown* data.

Table 1, Table 2 and Table 3 are simulation results of 9, 15 and 30 training data. Each tables has four categories based on which group is set to as unknown data. We set alcohol 0% as unknown data at number one, alcohol 15% at number two, alcohol 25% at number three and alcohol 35% at number four. From those tables, we can see that IPGNN can approximate well for unlearned data. The *unlearned* data is a data whose group is trained to network but the data is not. For example, we have 65 data of alcohol 25%, if we set 10 data as training data then the rest of 55 data is called as unlearned data. We also can see that IPGNN gives acceptable results of unlearned data for this case when the number of training data is set to 30. Therefore, we stop to increase the number of training data at this manner.

Table 1. Computer simulation for nine learning data

No	Unlearned Data	Unknown Data	GE
1	0.0046	0.0008	0.0027
2	0.0033	0.0193	0.0113
3	0.0013	0.0386	0.0200
4	0.0022	0.1225	0.0624

Table 2. Computer simulation for fifteen learning data

No	Unlearned Data	Unknown Data	GE
1	0.0031	0.0000	0.0016
2	0.0023	0.0180	0.0102
3	0.0001	0.0465	0.0233
4	0.0009	0.1224	0.0617

Table 3. Computer simulation for thirty learning data

No	Unlearned Data	Unknown Data	GE
1	0.0003	0.0000	0.0002
2	0.0002	0.0176	0.0089
3	0.0002	0.0550	0.0276
4	0.0002	0.1225	0.0614

Unfortunately, IPGNN still has not approximated well yet for unknown data while the other neural networks hasn't too [4]. This condition has impact on increasing generalization error (GE) of IPGNN. It is because GE is counted as average of error of unlearned and unknown data. To overcome this weakness, Kusumuputro at al is developing some approaches in both hardware and software system. One of them is increasing the number of sensor. This is expected to expand the interval of recognition space so that data become more compact and we can approximate unknown data based on the trained data.

Finally, we will compare generalization error of IPGNN is compared with other neural networks, i.e. resource allocating networks (RAN) and on-line back propagation (BP). Simulations are carried out in the following conditions:

- The number of learning data is set to 30. The simulation is divided into four categories based on which group of data is set to as unknown data.
- BP. The number of hidden units is choose manually for optimal generalization capability and fixed throughout the learning process.
- RAN. Parameters are assigned as $\varepsilon = 0.2$, $\delta_{max} = 0.7$, $\delta_{min} = 0.07$, $\kappa = 0.5$

Generalization error of the neural networks is shown in Table 4. This result shows that IPLNN provides better approximation than both BP and RAN. It also means that IPGNN can capture information about the percentage of alcohol faster than RAN. Meanwhile, IPGNN can provide better result than BP that is batch learning. The reason is due to learning results obtained by IPGNN are exactly the same as those obtained by its batch-learning version.

Table 4. Comparison of generalization error

No	IPGNN	RAN	BP
1	0.0002	0.0359	0.0217
2	0.0089	0.0170	0.0141
3	0.0276	0.0582	0.0307
4	0.0614	0.0797	0.0774

5 Concluding Remark

In this paper, we have adopted a two-level learning hierarchy for constructing IPGNN. The learning strategy is based on the combine GA and IPLNN algorithm. The orders of the reproducing kernel Hilbert space are optimized using the GA at the upper level. Given these parameters, the IPLNN algorithm is used to construct IPGNN at the lower level. The network produced by this learning hierarchy is applied as part of AOD system to approximate percentage of alcohol for small-size learning data.

Based on our computer simulation, we have results that IPGNN can approximate well for unlearned data. However, IPGNN still has not recognized well yet for unknown data. This condition has impact on reducing generalization error of IPGNN.

In comparison with other neural network, i.e. on-line back propagation networks (BP) and resource allocation network (RAN), We have results that IPGNN provides better generalization capability than two other neural networks.

References:

[1] S. Chen, C.F.N. Cowan, P. M. Grant, Combined Genetic Algorithm Optimization and Regularized Orthogonal Least Squares Learning for Radial Basis Function networks, *IEEE Transaction on Neural Networks*, Vol. 10, No. 5, 1999, pp. 1239-1243

[2] C. R. Houck, J. A. Jones, M. G. Kay, A Genetic Algorithm for Function Optimization: A Matlab Implementation, *Technical Report*, North Carolina State University, 1999

[3] V. Kadirkamanathan, M. Nirajan, A Function Estimation Approach to Sequential Learning with Neural Networks, *Neural Computation 5*, 1993, pp. 954-975

[4] B. Kusumoputro, M. Rivai, Discrimination of Fragrance Odor by Array Quartz Resonator and Neural Networks, *Computational Intelligence and Multimedia Application*, World Scientific Pubs. Co, 1998

[5] H. Murfi, B. Kusumoputro, Evaluation of Generalization Capability of Incremental Projection Learning-Based Neural networks, *Master Thesis*, Faculty of Computer Science, University of Indonesia, 2002

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

[7] J. Platt, A Resource Allocating Networks for Function Interpolation, *Neural Computation*, Vol. 3, No. 2, 1991, pp. 213-225

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

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

[10]M. Sugiyama, H. Ogawa, Incremental construction of projection generalizing neural networks, To Appear in *IEICE Transactions on Information and Systems*

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

[12]Y. Yingwei, N. Sundararajan, P. Saratchandran, A Sequential Learning Schema for Function Interpolation Using Minimal Radial Basis Function Networks, *Neural Computation*, Vol. 9, 1997, pp. 461-478