International Journal of Fuzzy Logic and Intelligent Systems, vol. 7, no. 2, June 2007 pp. 132-137

Hybrid Fuzzy Neural Networks by Means of Information Granulation and Genetic Optimization and Its Application to Software Process

Byoung-Jun Park*, Sung-Kwun Oh, and Young-Il Lee****

*** Department of Electrical Electronic and Information Engineering, Wonkwang University, 344-2, Shinyong-Dong, Iksan, Chon-Buk, 570-749, South Korea**

**** Department of Electrical Engineering, The University of Suwon, San 2-2, Wau-ri, Bongdam-eup, Hwaseong-si, Gyeonggi-do, 445-743, South Kore**

Abstract

Experimental software data capturing the essence of software projects (expressed e.g., in terms of their complexity and development time) have been a subject of intensive modeling. In this study, we introduce a new category of Hybrid Fuzzy Neural Networks (gHFNN) and discuss their comprehensive design methodology. The gHFNN architecture results from highly synergistic linkages between Fuzzy Neural Networks (FNN) and Polynomial Neural Networks (PNN). We develop a rule-based model consisting of a number of "if-then" statements whose antecedents are formed in the input space and linked with the consequents (conclusion parts) formed in the output space. In this framework, FNNs contribute to the formation of the premise part of the overall network structure of the gHFNN. The consequences of the rules are designed with the aid of genetically endowed PNNs. The experiments reported in this study deal with well-known software data such as the NASA dataset. In comparison with the previously discussed approaches, the proposed self-organizing networks are more accurate and yield significant generalization abilities.

Key Words : Hybrid Fuzzy Neural Networks(HFNN), Fuzzy neural networks(FNN), Polynomial neural networks(PNN), GAs, Evolutionary design, NASA software dataset

1. Introduction

Empirical studies in software engineering employ experimental data to gain insight into the software development processes and assess their quality. Data concerning software products and software processes are crucial to their better understanding and, in the sequel, the development of effective ways of producing high quality software. Efficient modeling techniques should allow for a selection of pertinent variables and a formation of highly representative datasets. The models should be able to take advantage of the existing domain knowledge and augment it by available numeric data to form a coherent data-knowledge modeling entity. One of the omnipresent modeling tendencies exploits techniques of Computational Intelligence (CI) cf.[1,2].

In this study, we develop a hybrid modeling architecture, called genetically optimized Hybrid Fuzzy Neural Networks (gHFNN). In a nutshell, gHFNN is composed of two main substructures driven by genetic optimization, namely a fuzzy set-based fuzzy neural network (FNN [3]) and a genetic polynomial neural network (gPNN). The role of the FNN is to interact with input data and granulate the corresponding input spaces. In the first case (referred to as Scheme I) we concentrate on the use of a simplified fuzzy inference. In the second case (Scheme II), we take advantage of the linear form of fuzzy inference. The role of the gPNN is to carry out a nonlinear transformation completed at the level of the fuzzy sets being formed at the level of FNNs. The gPNN exhibiting a flexible and versatile topology is constructed on a basis of Group Method of Data Handling (GMDH [4,5]) and in this development uses the mechanisms of genetic algorithms (GAs [6]). The design procedure applied to the construction of each layer of the PNN deals with its structural optimization involving the selection of optimal nodes (that is polynomial neurons; PNs) being equipped with some specific local characteristics (such as the number of input variables, the order of the polynomial, and a collection of the subsets of input variables). It also helps address specific aspects of the parametric optimization. To assess the performance of the proposed model, we experiment with NASA dataset [7] widely used in quantitative software engineering.

2. Conventional Hybrid Fuzzy Neural Networks (HFNN)

The architectures of conventional HFNNs [8,9] resulted through a synergy between two general constructs such as FNN and PNN. Based on the different topologies of PNN, the

Manuscript received Sep. 11, 2006; revised May. 22, 2007.
This work has been supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD)(KRF-2006-311-D00194)

HFNN distinguishes between two kinds of topologies of the models, namely basic and modified architectures. For each of these architectures we introduce their comprehensive taxonomy distinguishing between several detailed cases. The design of the HFNN is completed in a stepwise manner through a generation of a series of successive layers. Each layer consists of a certain number of nodes (PNs) for which the number of input variables could the same as in the previous layers or may differ across the network. The structure of the HFNN is selected on a basis of the number of input variables and the order of the polynomial allocated to each layer. We have

・ Basic: The number of input variables of PDs of PNN is kept the same in every layer.

・ Modified: The number of input variables of PDs of PNN is made variable in each layer.

- Case 1. The polynomial order of PDs of PNN is same in every layer.

- Case 2. The polynomial order of PDs in the 2nd layer or higher of PNN has a different or modified type in comparison with the one of PDs present in the 1st layer.

According to the alternative of two connection points, a certain combination of fuzzy set based FNN and PNN was considered for the formation of the HFNN architecture. Giventhe connection point, if input variables to PNN used on the consequence part of HFNN are less than three (or four), the generic type of the HFNN does not generate a highly versatile structure. Bearing this in mind, we identify the following types of structures

・ Generic type of HFNN; Combination of FNN and the generic PNN

・ Advanced type of HFNN; Combination of FNN and the advanced PNN

3. The architecture and design procedure of the gHFN

In a nutshell, a gHFNN emerges from the genetically optimized multi-layer perceptron architecture based on fuzzy set-based FNN, GAs and GMDH. In what follows, we start with a discussion of their functional components and then move on to the architectural considerations and finally elaborate on the overall design process.

3.1 Fuzzy neural networks and genetic optimizatio

We develop a FNN based on the two types of mechanisms of fuzzy inference, that is, a simplified one (referred to as Scheme I) and a linear fuzzy inference-based FNN (called Scheme II). The output of the FNN is governed by the following expression.

$$\hat{y}=f_1(x_1)+f_2(x_2)+\cdots+f_m(x_m)=\sum_{i=1}^{m}f_i(x_i) \tag{1}$$

We can regard each fi(xi) given by (1) as the following mappings (rules).

Scheme I -Rj : If xi is Aij then Cyij=wij (2)

Scheme II -Rj : If xi is Aij then Cyij=wsij + wij xi (3)

Rj is the j-th fuzzy rule while Aij denotes a fuzzy variable of the premise of the fuzzy rule and is represented by a membership function μij. wsij and wij stand for the connections between the corresponding neurons as visualized in Fig. 1. The mapping from xi to fi(xi) in (2) is determined by fuzzy inferencing that is followed by a standard defuzzification step that is

$$f_i(x_i)=\sum_{j=1}^{z}\mu_{ij}(x_i)\cdot w_{ij}\Big/\sum_{j=1}^{z}\mu_{ij}(x_i) \tag{4}$$

The learning of FNN is realized by adjusting connections of the neurons and as such it follows a standard algorithm of Back-Propagation (BP). For the simplified fuzzy inference-based FNN, the update formula of the corresponding connection occurring in Scheme I reads in the form

$$\Delta w_{ij}=2\cdot\eta\cdot(y_p-\hat{y}_p)\cdot\mu_{ij}(x_i)+\alpha(w_{ij}(t)-w_{ij}(t-1)) \tag{5}$$

In the above expression, yp is the p-th target output data, $\hat{y}_p$ stands for the p-th actual output of the model for this specific data point, η is a positive learning rate and α is a momentum coefficient constrained to the unit interval. The inference result and the learning algorithm in linear fuzzy inference-based FNN use the mechanisms in the same manner as discussed above. In order to enhance the learning of the FNN and augment its performance of a FNN, we use GAs to adjust learning rate, momentum coefficient and the parameters of the membership functions of the antecedents of the rules.

The FNN structure exhibits two possible connection points. Its location implies the character of the network (viz. its flexibility and learning capabilities). Note that the first connection point allows perceiving each linguistic manifestation of the original variables (viz. these variables are transformed by fuzzy sets and normalized). The location of the other connection point implies that the PNN part of the network does not "see" the individual fuzzy sets being located in the input space.

3.2 Genetically optimized PNN (gPNN)

Being cognizant of possible drawbacks of gradient-based optimization techniques, we augment the design process by the mechanisms of genetic optimization and genetic algorithms, in particular.

When we construct PNs located at each layer of the conventional PNN [4], their parameters such as the number of input variables, the order of polynomial, and the number of input variables available within a PN are fixed in advance by the designer. This could have frequently contributed to the difficulties in the design of the optimal network. To overcome this apparent drawback, we introduce a new genetic design approach. As a consequence we will be referring to these networks as genetic PNN (gPNN" for brief). The essence of the genetically supported optimization process of PNN is shown in Fig. 1. The determination of the optimal values of the parameters available within an individual PN leads to a structurally and parametrically optimized network. As a result, this network is more flexible anc exhibits simpler topology in comparison to the conventioral PNN discussed in the previous research.

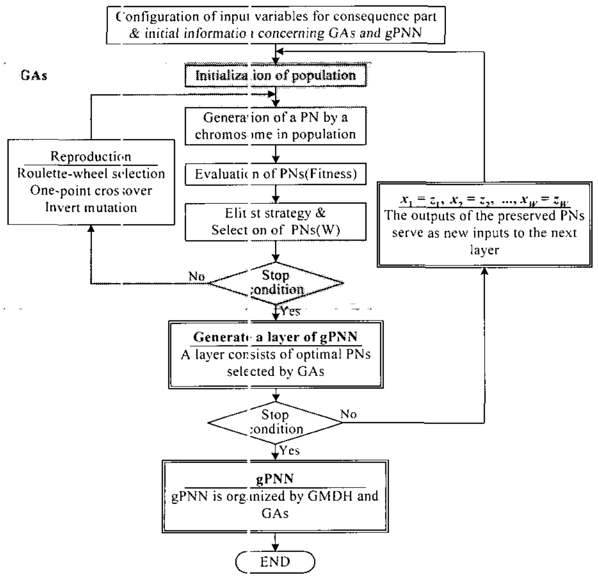

Configuration of inpu variables for consequence part & initial informatio concerning GAs and gPNN
GAs
Initializa ion of population
Genera ion of a PN by a chromos me in population
Reproduction
Roulette-wheel selection
One-point cros over
Invert mutation
Evaluatic n of PNs(Fitness)
Elit st strategy & Select on of PNs(W)
$x_1 = z_1, x_2 = z_2, \ldots, x_W = z_W$
The outputs of the preserved PNs serve as new inputs to the next layer
No
Stop condition
Yes
Generate a layer of gPNN
A layer consists of optimal PNs selected by GAs
Stop condition
No
Yes
gPNN
gPNN is organized by GMDH and GAs
END

Fig. 1.Overall genetically-driven optimization process of gPNN

3.3 The algorithms and design procedure of gHFNN

Taking into account the underlying mechanisms of evolutionary and gradient-based optimization, we organize the entire design in a systematic fashion. In what follows, we summarize the overall architecture and the ensuing design steps.

The premise of gHFNN: FNN (Refer to Fig. 1)

[Layer 1] Input layer. The role of this layer is to distribute the signals to the nodes located at the next layer.

[Layer 2] Computing activation degrees of linguistic labels. Each node in this layer corresponds to one linguistic label (small, large, etc.) of the input variables in layer 1.

[Layer 3] Normalization of a degree activation (firing) of the rule.

[Layer 4] Multiplication of a normalized activation degree of the rule by its corresponding connection. The calculated activation degree at the thirc layer is now calibrated through the values of the connections, namely

$$a_{ij} = \bar{\mu}_{ij} \times Cy_{ij} = \mu_{ij} \times Cy_{ij}, \quad \begin{cases} \text{Simplified} : Cy_{ij} = w_{ij} \\ \text{Linear} \quad : Cy_{ij} = ws_{ij} + w_{ij} \cdot x_i \end{cases} \tag{6}$$

If we choose Connection point 1 for combining FNN with gPNN as shown in Fig. 1, aij is regarded as the input variable of the gPNN.

[Layer 5] Fuzzy inference for the fuzzy rules. If we choose Connection point 2, fi is the input variable of gPNN.

[Layer 6; Output layer of FNN] Computing output of a FNN. The output becomes a sum of the individual contributions from the previous layer as shown in eq. (1).

The consequence of gHFNN: gPNN (Refer to Fig. 2)

[Step 1] Configuration of input variables. If we choose the first option (Connection point 1), x1=a11, x2=a12,···, xn=aij (n=i×j). For the second option (Connection point 2), we have x1=f1, x2=f2,···, xn=fm (n=m).

[Step 2]Setting up initial configuration required for the construction of the gPNN. We decide upon the design parameters of the gPNN structure; ultimately they shape up the optimization environment and required computational effort.

[Step 3] Initialization of population.

[Step 4] Development of PNs structure through genetic optimization. Refer to PN related polynomial type of Table 1.

Table 1. Different forms of regression polynomial used in the PN

Number of inputs / Order of the polynominal	2	3	4
1 (Type 1)	Bilinear	Trilinear	Tetralinear
2 (Type 2)	Biquadratic-1	Triquadratic-1	Tetraquadratic-1
3 (Type 3)	Biquadratic-2	Triquadratic-2	Tetraquadratic-2

[Step 5] Evaluation of PNs. To evaluate the performance of PNs (nodes) constructed in each population, we use a certain objective function (Performance Index; PI) which in turn is used to form the corresponding fitness function. Typically the fitness is taken as an inverse of the objective function so the decrease in the performance index leads to the increase of the values of the fitness function. The typical fitness function comes in the form (note that the absolute differences standing in this sum contribute to the increased robustness of the resulting model)

$$E(PI \text{ or } EPI) = \frac{1}{n}\sum_{p=1}^{n}\frac{|y_p - \hat{y}_p|}{y_p} \tag{7}$$

[Step 6] Elitist strategy and selection of PNs with the best predictive capability. We select the values of W of PNs characterized by the best fitness values.

[Step 7] Reproduction. To move on to the next generation, we carry out selection, crossover, and mutation operations using genetic information and the fitness values obtained via Step 5.

[Step 8] Repetition of Steps 4 through 7. The iterative process generates the optimal nodes of the given layer of the gPNN.

[Step 9] Construction of the corresponding layer.

[Step 10] Verification of the satisfaction of the termination criterion.

[Step 11] Specification of new input variables to be used in the next layer of the network. The outputs of the retained nodes (zli, z2i, ···, zWi) serve as new inputs to the next layer of the network (x1j, x2j, ···, xWj) (j=i+1, i and j note ith and jth layer, respectively).

The overall design of the gPNN is completed by repeating a sequence of steps 4-11 presented above

3.4 Model selection

Typically,any model selection procedure is based on seeking a sound compromise between approximation and generalization errors. The main performance measure that we use here is the MMRE (the mean magnitude of relative error) as expressed by (7). For evaluation of the generalization abilities of the model, many estimates have been proposed in the literature. The most popular ones are the holdout estimate and the k-fold cross-validation estimate [10]. The holdout estimate is obtained by partitioning the dataset into two mutually exclusive subsets called training and test sets. The error estimate used on the test set is considered to assess the generalization abilities of the constructed model. On the other hand, the k-fold cross-validation estimate is obtained by a sample reuse technique. The dataset is divided into "k" mutually exclusive subsets of (almost) equal size, k-1 subsets are used for training, and the kth is used for prediction. This process is repeated "k"times, each time employing a different subset for assessing the prediction of the model.

When the value of "k"is equal to the size of the overall data, the resulting technique is referred to as aleave-one-out cross-validation (LOOCV) estimate. In this study, we employ the LOOCV estimate of generalization error. There are two essential reasons behind this selection. First, this estimate comes with useful statistical properties and helps avoid a potential bias [11]. Second, it seems to be particularly suited for small size datasets [12].

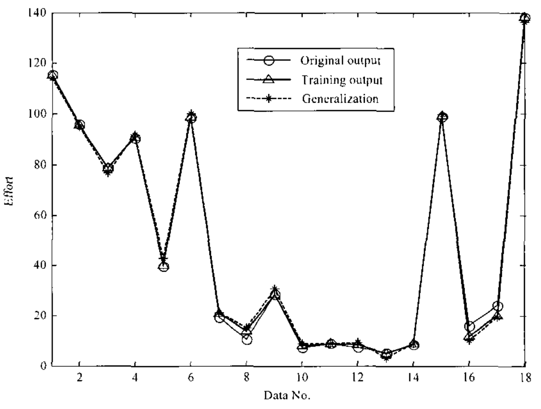
4. Experimental studies

The experimental studies are concerned with a software effort dataset coming from NASA [7]. The dataset involves two independent variables, viz. Developed Lines of Code (DL) and used Methodology (ME). The output variable, Effort (Y), is expessed in man-months. For pertinent details refer to [7]. The comprehensive series of experiments involved various topologies of gHFNNs when considering only a single input namely the number of the developed lines of code (which seems to be a sound input variable).

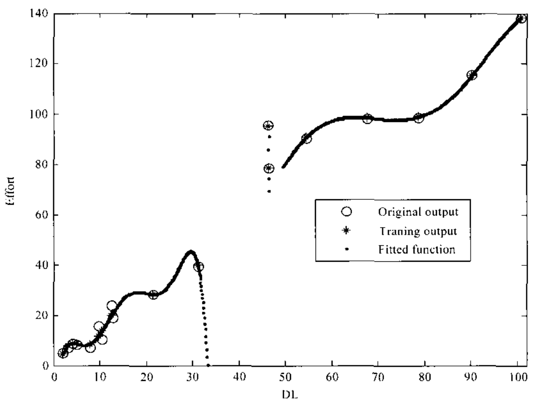
Table 2. Performance index of gHFNN of a single-input system (with the DL being the input variable)

Premise part				CP	Consequence part						PI	EPI
Fuzzy Inference	No. of rules(MFs)	PI	EPI		Layer	No. of inputs	Input No.			T		
Simplified	2 (2)	0.164	0.180	01	1	1	1			1	0.210	0.232
					2	3	3	4	7	3	0.118	0.202
					3	2	7	14		2	0.105	0.141
Linear	2 (2)	0.145	0.166	01	1	2	1	2		3	0.135	0.167
					2	3	1	53	7	3	0.106	0.153
					3	3	6	13	30	2	0.0747	0.131

Table 2 summarizes the results of the optimized architectures according to connection points based on each fuzzy inference method. We distinguish between two architectures such as the premise FNN and the overall gHFNN. First, in the case of the premise FNN, the network comes in the form of two fuzzy inference methods. Here, the FNN uses two membership functions for input variable and has two fuzzy rules. In this case, as mentioned previously, the parameters of the FNN are optimized with the aid of GAs and BP learning. When considering the simplified fuzzy inference-based FNN, the minimal value of the performance index, that is PI= 0.164 and EPI=0.180 are obtained. In the case of the linear fuzzy inference-based FNN, the best results give ise to the values of the performance index equal to PI=0.145 and EPI=0.166. The values of the performance index ofthe gHFNN depend on the connection point based on the individual fuzzy inference methods. The values of the performance index vis-à-vis the considered number of layers of the gHFNN and related to the optimized architectures occurring at each layer of the network are shown in Table 2. For example, let us investigate the 3rd layer of the network in the case of the linear fuzzy inference and connection point 1. The fitness value in layer 3 is maximal when nodes 6, 13, and 30 (such as z6, z13, and z30) are selected as those preferred in the previous layer (viz. the 2nd layer). Referring to Table 2, the blank entries denoted by (.) indicate that the node has not been selected by the genetic optimization. Finally, the optimal network-related input-output characteristics are illustrated in Fig. 2.



(a)



(b)

Fig. 2. Input-output characteristics of the optimal networks; (a) Original output and model output for training and generalization (b) Plot of the fit of the gHFNN model to the data treated as a function of DL.

Now we develop the software effort estimation model based on two independent variables that is DL and ME. Following the same design process and considering both the simplified and linear fuzzy inference based FNN, the values of the performance index are summarized in Table 3.

Table 3. Performance index of gHFNN for two input network (DL, ME); shown are various topologies investigated in the design process

Premise part				CP	Consequence part								
Fuzzy Inference	No. of rules (MFs)	PI	EPI		Layer	No. of inputs	Input No.				T	PI	EPI
Simplified	4 (2+2)	0.102	0.114	01	1	2	1	4			3	0.0908	0.118
					2	2	21	29			1	0.0829	0.0984
					3	4	15	22	25	30	3	0.0407	0.0896
				02	1	2	1	2			3	0.0908	0.118
					2	2	5	6			1	0.0829	0.0985
					3	1	25				1	0.0706	0.0799
Linear	4 (2+2)	0.0959	0.119	01	1	3	1	2	4		1	0.117	0.154
					2	4	13	16	21	30	2	0.0117	0.0723
					3	4	2	4	13	17	2	0.00014	0.00307
				02	1	2	1	2			2	0.0916	0.133
					2	2	2	5			2	0.0687	0.0950
					3	3	3	7	14	.	3	0.0454	0.0719

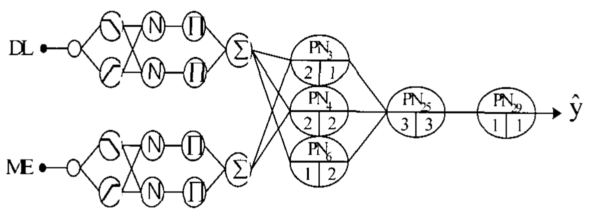


Fig. 3. Optimal topology of gHFNN for 2 system inputs (Simplified, connection 02)

When considering the simplified fuzzy inference-based FNN with four fuzzy rules, the best performance (PI= 0.0706 and EPI=0.0799) is reported when using the connection point 02. In the case of the linear fuzzy inference-based FNN with four fuzzy rules, the best results (PI=0.00014 and EPI=0.00307) when the connection point is set up as 01.

The optimal topologies of gHFNN for 3 layers of gPNN are shown in Fig. 3. In each node of Fig. 3, 'PNn' denotes the nth PN (node) of the corresponding layer, the number shown at the lower left side denotes the number of nodes (inputs or PNs) entering the corresponding node. The number at the lower right side indicates the polynomial order used at the corresponding node. The best results come with the performance index where PI=0.0706 and EPI=0.0799. The characteristics of the network are illustrated in Fig. 4 and Fig. 5.

Table 4 contrasts the performance of the genetically developed network with other fuzzy-neural networks and conventional HFNN already available in the literature.

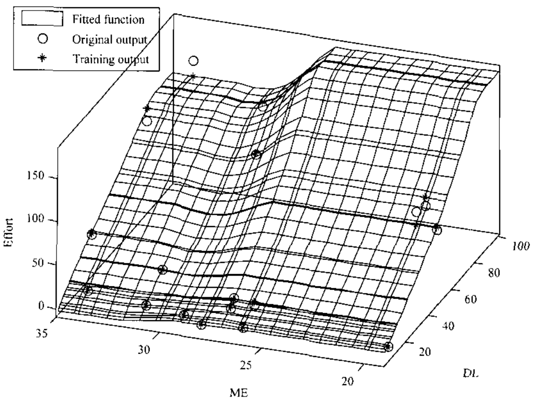


Fig. 4. 3-D plot of the output characteristics of the two-input gHFNN (simplified inference, connection 02)

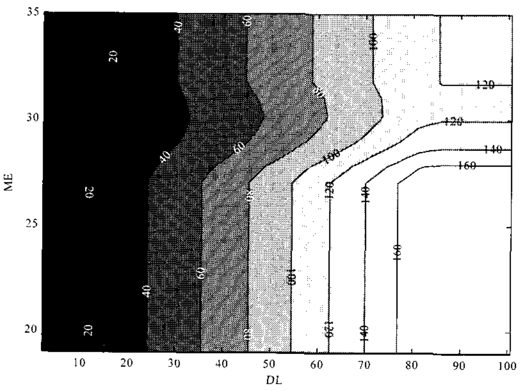


Fig. 5. Contour plot of the output characteristics of the gHFNN (simplified, connection 02)

Table 4. Performance analysis of selected models used for the NASA software data

Model			System inputs	PI	EPI	No. of rules
Shin and Goel's RBF model[7]			DL	0.1579	0.1870	3 rules
				0.1450	0.1881	6 rules
			DL,ME	0.1470	0.2474	4 rules
				0.0870	0.1907	7 rules
Conventional HFNN	FS based[8]	Simplified	DL,ME	0.0886	0.0907	4rules/3th layer
	FR based[9]	Simplified	DL	0.175	0.185	2rules/2th layer
			DL,ME	0.0493	0.0565	4rules/2th layer
		Linear	DL	0.0877	0.131	2rules/3th layer
			DL,ME	0.0630	0.0736	4rules/3th layer
Proposed model	FNN	Simplified	DL	0.164	0.180	2 rules
			DL,ME	0.102	0.114	4 rules
		Linear	DL	0.145	0.166	2 rules
			DL,ME	0.0959	0.119	4 rules
	gHFNN	Simplified	DL	0.105	0.141	2rules/3th layer
			DL,ME	0.0407	0.0896	4rules/3th layer
		Linear	DL	0.0747	0.131	2rules/3th layer
			DL,ME	0.00014	0.00307	4rules/3th layer

5. Conclusion

The comprehensive design methodology supports both structural and parametric optimization of the networks. Considering the premise structure of the gHFNN, the optimization of the rule-based FNN hinges on the use of genetic optimization and the back-propagation (BP) learning algorithm: The GAs leads to the auto-tuning of vertexes of membership function, while the BP algorithm helps produce the optimal values of the parameters of the consequent polynomial of fuzzy rules. The gPNN that is the consequent structure of the gHFNN is based on the mechanisms of the extended GMDH and GAs. The extended GMDH starts with a phase of structural optimization (such as a self-organizing and evolutionary algorithm) that is followed by the parametric optimization (least square estimation). The gPNN architecture is driven by the genetic optimization, which results in the selection of optimal nodes. We also discussed a variety of architectures of the gHFNN. Modeling software data realized in this framework has led to useful results and demonstrated the usefulness of the discussed models. In particular the experimental studies concerning NASA data demonstrate that the gHFNNs produce better results than some other models of quantitative software engineering.

References

[1] W. Pedrycz: Computational Intelligence. (1998) CRC Press, FL.

[2] W. Pedrycz, J.F.Peters: Computational Intelligence in Software Engineering. (1998) Singapore: World Scientific Publishing Co. Pte. Ltd.

[3] S.-K. Oh, W. Pedrycz, H.-S. Park: Hybrid Identification in Fuzzy-Neural Networks. Fuzzy Sets and Systems, 138(2) (2003) 399-426

[4] S.-K. Oh, W. Pedrycz, B.-J. Park: Polynomial Neural Networks Architecture: Analysis and Design. Computers and Electrical Engineering, 29(6) (2003) 653-725

[5] A. G. Ivahnenko: The group method of data handling: a rival of method of stochastic approximation. Soviet Automatic Control, 13(3) (1968) 43-55

[6] Z. Michalewicz: Genetic Algorithms + Data Structures = Evolution Programs. (1996) Springer-Verlag, Berlin Heidelberg

[7] M. Shin, A.L. Goel: Empirical Data Modeling in Software Engineering Using Radial Basis Functions. IEEE Trans on Software Engineering, 26(6) (2000) 567-576

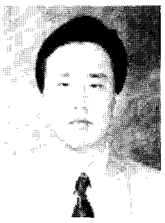
[8] S.-K. Oh, W. Pedrycz , B.-J. Park: Self-organizing neurofuzzy networks in modeling software data. Fuzzy set and systems, 145(1) (2004) 165-181

[9] S.-K. Oh, W. Pedrycz , B.-J. Park: Relation-based Neurofuzzy Networks with Evolutionary Data Granulation. Mathematical and Computer Modelling, 40(7-8) (2004) 891-921

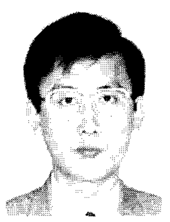
[10] C.M. Bishop: Neural Networks for Pattern Recognition. (1995) Oxford Univ. Press

[11] M. Kearns, D. Ron: Algorithmic Stability and Sanity-Check Bounds for Leave-One-Out Cross-Validation. Proc. 10th Ann. Conf. Computational Learning Theory, (1997) 152-162

[12] C.F. Kemerer: An Empirical Validation of Software Cost Estimation Models. Comm. ACM, 30(5) (1987) 416-429



Byoung-Jun Park received the B.S., M.S., and PhD. degrees in control and instrumentation engineering from Wonkwang University, Korea in 1998, 2000 and 2003, respectively. He currently holds a position of a Postdoctoral Fellow in the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada. His research interests include fuzzy, neurofuzzy systems, genetic algorithms, Computational Intelligence, hybrid systems, and intelligent control.



Sung-Kwun Oh received the BSc, MSc, and PhD. degrees in Electrical Engineering from Yonsei University, Seoul, Korea, in 1981, 1983, and 1993, respectively. During 1983-1989, he was a Senior Researcher of R&D Lab. of Lucky-Goldstar Industrial Systems Co., Ltd. From 1996 to 1997, he held a position of a Postdoctoral fellow in the Department of Electrical and Computer Engineering, University of Manitoba, Canada. He is currently a Professor in the Department of Electrical Engineering, University of Suwon, South Korea. His research interests include fuzzy system, fuzzy-neural networks, automation systems, advanced Computational Intelligence, and intelligent control. He is a member of IEEE. He currently serves as an Associate Editor of KIEE Transactions on Systems & Control, International Journal of Fuzzy Logic and Intelligent Systems of the KFIS, and International Journal of Control, Automation, and Systems Engineering of the ICASE, South Korea. E-mail: ohsk@suwon.ac.kr

Young-Il Lee He received the B.S. degree in Electronic Engineering from Yonsei University, Seoul, Korea in 1980. From 1979 to 1987 he worked, in sequence, as a computer H/W engineer at Oricom Inc., a computer S/W engineer at Daehan Tele. Co., a test engineer at Ericfon Co., and a chief engineer of the AV center at Oricom Inc.. He resumed his studies in 1989 and received the M.S. and Ph.D. degrees in Electrical Engineering from Purdue University, West Lafayette, U,S.A. in 1990 and 1996, respectively. He is currently an Assistant Professor in Electrical Engineering, Suwon University, Korea. He is a member of KIEE and IEEE. His research interests include the fields of power system, electronic devices, and fuzzy system.