LETTER

Interpretable Neural Network Tree for Continuous-Feature Data Sets

Qinzhen Xu
Depterment of Radio Engineering, Southeast University, Nanjing, Jiangsu,China,210096
E-mail: summer@seu.edu.cn

Qiangfu Zhao
University of Aizu, Aizu-wakamatsu, 965-8580 Japan
Email: qf-zhao@u-aizu.ac.jp

Wenjiang Pei*,Luxi Yang, Zhenya He
Depterment of Radio Engineering, Southeast University, Nanjing, Jiangsu,China,210096
*E-mail: wjpei@seu.edu.cn

(Submitted on March 15, 2004)

***Abstract*—**Neural network tree (NNTree) is a hybrid learning model. Currently We have proposed a simplified multiple objective optimization based genetic algorithm for evolving such kind of NNTrees and shown through experments that an NNTree can be interpreted easily if the number of inputs for each expert neural network (ENN) is limited. One problem remained is that for those problems the input features are continuous. This means that even if the number of inputs for each ENN is, say 4, the number of corresponding binary inputs will be 64 if each continuous input is represented with a 16-bit binary number. This also means that the computational complexity is proportional to 2^64. To make the NNTrees more interpretable, we propose an interpretable NNTree through self-organized learning of features. We will show through experiments that the NNTrees built from the training data after self-organized learning are equally good as those obtained from the original data. Further, the number of quantization points in each dimension is usually less than 10 for the databases we used. This means that 3 or 4 binary inputs are enough to represent each continuous input, and thus, the NNTrees so obtained are much more interpretable.

***Keywords*—**Neural network tree, Multiple objective optimization, Genetic algorithm, Computational complexity, self-organized learning

1. Introduction

Algorithms in machine learning can be roughly divided into the two categories of symbolic approaches and non-symbolic ones. Symbolic approaches, such as Decision Tree (DT), are generally considered as comprehensible but not suitable for on-line learning. On the contrary, non-symbolic ones, such as neural network, can adapt to learn in changing environment while it is always incomprehensible due to its black-box learning process.

To have the advantages of both symbolic and non-symbolic approaches, it is required in many situations that machine learning algorithm should be both comprehensible and learnable on-line. For this purpose, we designed Neural Network Tree (NNTree) [1]. An NNTree is actually a modular neural network with the overall structure being a decision tree (DT), and each non-terminal node being an expert neural network (ENN). It has been proved by experiment of digit recognition that NNTree is more efficient than traditional decision tree in the sense that higher recognition rate can be achieved with less nodes. Thus, with the designed NNTree, a comprehensible result form and refined knowledge all together with a single model can be obtained.

Research of simplifying the interpretation of NNTree has been conducted currently [2][3]. The computational complexity of extracting comprehensible rules from a neural network (NN) is usually exponential [4]. For each trained ENN of an NNTree, the time complexity for interpreting increases exponentially with the

number of inputs. NNTrees with nodes of limited number of inputs was studied to achieve interpreting learned knowledge in polynomial time [2]. To make the results as simple as possible, we have also proposed a multiple objective optimization based genetic algorithm (MOO-GA) for designing NNTrees that are both interpretable and comprehensible [5]. Here, "interpretable" means that the NNTrees can be interpreted easily (say, in polynomial time), and "comprehensible" means that the rules extracted from the NNTrees are easily understandable, even by human users.

One problem remained is that for many problems the input features are continuous. This means that even if the number of inputs for each ENN is, say 4, the number of corresponding binary inputs will be 64 if each continuous input is represented with a 16-bit binary number. This also means that the computational complexity (computational time and memory space) is proportional to 2^64. To lighten the computational complexity, especially the spatial complexity, we propose to quantize the continuous inputs using self-organized learning in each dimension. We will show through experiments that the NNTrees built from the quantized training data are equally good as those obtained from the original data. Further, the NNTrees so obtained are much more interpretable.

The organization of this paper is as follows. After the introduction, we describe the designing of NNTree in detailed in section 2. In section 3, we focus on the problem of computational complexity and propose the process of self-organized learning of features. Experimental results are presented to confirm the proposed approach in section 4. Eventually, we summarize the paper in section 5.

2. Designing of NNTree

3.1 NNTree Structure

We adopt the structure of NNTrees designed in [1], which is to embed NNs directly into DTs at each non-terminal node. Figure 1. is one of the structure examples. In this NNTree, each node is an expert neural network, which in our study, is a multiplayer perceptron (MLP) with one hidden layer and n (n is always 2 in our study) output neurons.

The basic idea is to design small ENNs first for extracting certain features and making local decision, and then put them together to get the whole decision tree. Free parameters contained in the ENNs can be updated to adapt to changing environment, while the global decision rules keep unchanged. It is possible to reduce the tree size because each node is more powerful.

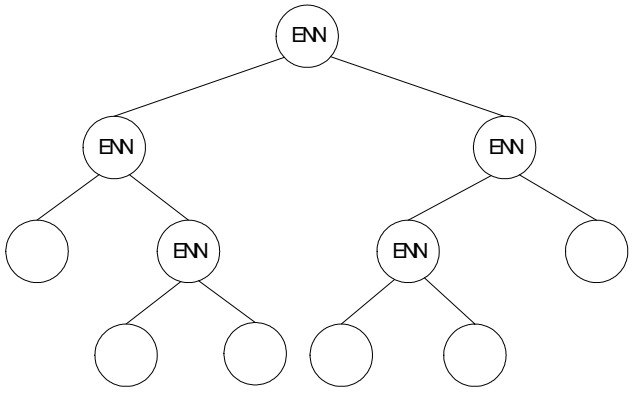


Figure 1. A structure of an NNTree

3.2 Design Of NNTRees

To construct a DT, it is often assumed that a training set consisting of feature vectors and their corresponding class labels are available. The DT is then constructed by partitioning the feature space in such a way as to recursively generate the tree. This procedure involves three steps: splitting nodes, determining which nodes are terminal nodes, and assigning class labels to terminal nodes.

The basic idea is to partition the training examples assigned to the current node in such a way that the average information required to classify a given example could be reduced most. The information gain ratio of

each features is test for the selection of the features for different nodes. The best-information-gain-ratio feature is selected and tested at the root node of the DT. A descendent of the root node is then split for each valuable of this feature, and the training example are then sorted to the appropriated descendent node. The entire process is then repeated using the training examples associated with each descendent node to select the best-information-gain-ratio feature to test at that node in the tree [6].

The information gain ratio is defined in terms of entropy. Let S stands for the set of all training examples assigned to the current node (with $|S|$ examples), and n_i the number of examples belonging to the $i-th$ class ($i = 1,2,...,N$), the average information (entropy) needed to identify the class of a given example is

$$\text{info}(S) = -\sum_{i=1}^{N} \frac{n_i}{|S|} \times \log_2(\frac{n_i}{|S|}) \tag{1}$$

Now suppose that S is partitioned into n sub-sets $S_1, S_2, \cdots, S_n$, by some test X, the information gain is given by

$$\text{gain}(X) = \text{info}(S) - \text{info}_X(S) \tag{2}$$

where

$$\text{info}_X(S) = \sum_{j=1}^{n} \frac{|S_i|}{|S|} \times \text{info}(S_i) \tag{3}$$

The information gain ratio is defined as follows:

$$\text{gain\_ratio}(X) = \text{gain}(X) / \text{split\_info}(X) \tag{4}$$

where

$$\text{split\_info}(X) = -\sum_{j=1}^{n} \frac{|S_i|}{|S|} \times \log_2(\frac{|S_i|}{|S|}) \tag{5}$$

For detailed discussion, one can refer to [7].

The overall process for designing an NNTree is the same as that for designing a DT. The only difference is to design an ENN for each non-terminal node. Since we do not know in advance which example should be assigned to which group, the only efficient way for designing the ENNs seems to be evolutionary algorithms (EAs). The information gain ratio is taken as the fitness function for the ENNs' evolution process.

To make the NNTrees interpretable, we can limit the number of inputs for each non-terminal node [2]. To make the extracted results more comprehensible, we need to reduce the number of hidden neurons [5]. Thus, for each non-terminal node, we need to firstly increase its partitioning ability (measured by information gain ratio), then decrease the number of inputs, and eventually decrease the number of hidden neurons.

It means that three objectives need to be optimized, and it is natural to use multiple objective optimization based GAs (MOO-GAs). One general approach involves the use of aggregating functions, such as *weighted sum* method, reduction to a single objective method, goal-attainment method and penalty function method. These methods will normally produce only the min-max optimum, but not the Pareto front unless a lot of weight combinations are tried. Non-Pareto approaches include Vector Evaluated Genetic Algorithm (VEGA) method, Lexicographic-ordering method, and Evolutionary strategies method. Pareto-based approaches include Pareto-based fitness assignment, Multiple Objective Genetic Algorithm (MOGA), Non-dominated Sorting Genetic Algorithm (NSGA), and Niched Pareto GA [8]. A sharing mechanism is needed in Pareto-based approaches to achieve good performance, and thus the computational cost is usually very large.

In our study, we introduce a simple MOO-GA [5]. The algorithm is a modified version of Goldberg's method, which is a kind of Pareto-ranking approach [11]. The modification is to sort the individual ENNs again according to their information gain ratio, if they have the same rank. The modified algorithm can provide one (the best individual) unique solution, rather than a set of non-dominating solutions. This is a simple method for automatic selection of the best solutions. It is also reasonable because we usually assign a higher priority to individuals with larger information gain ratio, or better partitioning ability.

Note that it is very important to keep the partitioning ability of the ENNs while trying to reduce the number of hidden neurons and the number of inputs. If the partitioning ability is reduced, the tree can be very large, even if each ENN is small. Such kinds of NNTrees are no more comprehensible. If we use ENNs with proper number of hidden neurons and proper number of inputs, the NNTree can be learnable, reliable, interpretable and comprehensible.

3. Self-organized Learning of Feature

For many problems, when the input features are continuous, the time and spatial computational complexity can be very high. At the same time the results are more incomprehensible. To reduce the complexity and make the result easily understandable, we propose to quantize the continuous inputs using self-organized learning in each dimension.

Since the features in each dimension are not always uniform distributed, non-uniform quantization may reduce the number of discrete values greatly.

We have tried to draw the class labels in one dimension and quantize the feature according to the number of change points. However, it is invalid due to the continually change of the class labels. That is to say, there are still large numbers of feature points after quantization. Further the NNTrees built from the so quantized dataset are not so good as the ones built from the original dataset.

In this paper, we quantize the feature of the datasets in each dimension in two steps. Firstly, we draw the histogram at each regularized feature dimension and decide the minimum quantization number of points according to the distribution of the peaks in the histogram. The minimum number of quantization points can decided by experience at present work. Once the minimum number m of quantization points is decided, the next step is to obtain the quantized values of the continuous features through self-organized learning of the features. The approach of Winner-Take-All (WTA) was adopted in the self-organized learning process for its simplicity and availability[9,10]. Numerous neural networks and circuits have been developed to perform the WTA competition [11]-[14]. In our research, we are interested in coding the continuous features into several representative values through WTA. Given any x belong to some continuous feature at one dimension; its representative feature points $w = (w_1, w_2, \cdots w_m)$ can be obtained through the following iteration steps:

step 1. Initialize representative feature points w randomly at interval (0,1). Set modification coefficient $\alpha = \alpha_0$ and maximum iteration time T, respectively.

step 2. Find the winner w_i, which is the nearest to x, that is to say,

$$|x - w_i| = \min_{1 \le j \le m} |x - w_j| \tag{6}$$

step 3. Modify w_i as follows:

$$w_i = w_i + \alpha(x - w_i) \tag{7}$$

where

$$\alpha = \alpha - \alpha_0 / T \tag{8}$$

Keep other w_j ($j = 1,2,\cdots m, j \neq i$) unchanged. Go through all the feature values x at one dimension. Record iteration times t.

step 4. If $t < T$, go to step 2. Else terminate the iteration.

When representative feature points at each dimension are determined, NNTree is trained on the quantized training sets.

4. Experiments and Results

Experiments were performed to determine the efficiency of NNTree after self-organized learning of features. Four datasets, Dermatology, E.coli, Ionosphere and BUPA (Liver) are involved in the experiments. The parameters of these datasets are enumerated in Table 1. They are all attainable from machine learning repository of the University of California at Irvine.

Main parameters taken in NNTree training and test experiments are listed as follows:

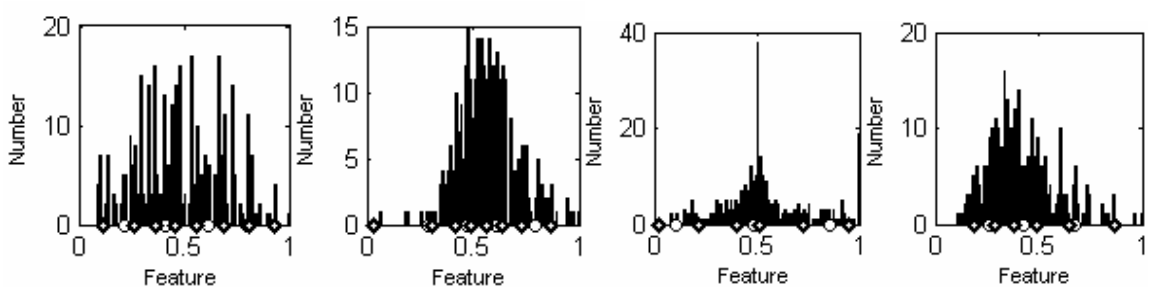
1). Number of generations: 1,000
2). Number of runs: 40
3). Population size: 100
4). Bit per weight: 16
5). Bit per feature position: 6 (Dermatology and ionosphere) and 3 (E.coli and BUPA)
6). Dynamic range of each weight: [-16.384, 16.384]

7). Selection rate: 0.2 (truncation selection)
8). Mutation rate: 0.01 (bit-by-bit)
9). Crossover rate: 1.0
10). Maximum iteration times for WTA: 100

Table 1.The Parameters of the Four Datasets

Datasets	Feature Number	Instance Number	Class Number	Continuous Features
Dematology	34	358	6	The 34th feature
E.coli	7	336	8	Except the 3rd and 4th features
Ionosphere	34	351	2	All the features
Bupa(Liver)	6	345	2	All the features

The process of self-organized learning of features was firstly conducted. Figure 2.(a)-(d) are some examples of the quantization results on the four datasets. The minimum number of quantization points is $m = 4$ for Dermatology and E.coli, and $m = 3$ for Ionosphere and Liver. They are all marked by "**o**" in Figure 2. $2m$ quantization points (i.e. double minimum number of quantization points) were also attempted in the experiments, and "◇" are the locations of representative points on normalized axial.



(a)Histogram of the 34th feature of dermatology (b) Histogram of the 5th feature of E.coli (c) Histogram of the 18th feature of Ionosphere (d) Histogram of the 2nd feature of Liver

Figure 2. Some quantization results on the four datasets.

When the representative points were determined, NNTrees were trained and tested on the quantized datasets. Figure 3. is one simple example of NNTrees constructed on quantized dataset of Ionosphere, where *F, H, C, L, IN* denote the used feature, number of hidden neurons, class distribution, class label, number of instances. The recognition rate of NNTree is 91.45%.

Table 2. to Table 5. are the comparison result of MOO-GA based NNTrees constructed on original datasets and the proposed NNTrees constructed on quantized datasets of the four datasets, respectively. Five items were considered in the experiments, which are size of DT (*Size*), number of hidden neurons of each ENN (N_H), recognition rate on training set (R_{train}), recognition rate on test set (R_{test}), and time consuming (*Time*).

We can get some heuristic indications from the four tables. Firstly, the most important point is that NNTrees constructed on the quantized datasets obtained a recognition rate very close to that constructed on original dataset on both training sets and test sets for most of the four datasets. That is also to say that NNTrees constructed on quantized datasets (which are far more compact than the original ones) are as good as those constructed on original ones as far as recognition rate are concerned.

Secondly, when the datasets quantized to m number of points, the tree size and hidden neuron number could be lager than that constructed on the original datasets. When the datasets were quantized to double-m number of points, *Size* and N_H came to very close to that of the original datasets. It means that the number of quantization points in each dimension is usually less than 10 for the databases we used. This means that 3 or 4 binary inputs are enough to represent each continuous input, and thus, the NNTrees so obtained can reduce the spatial computational complexity from 2^64 to almost 2^12 or 2^16.

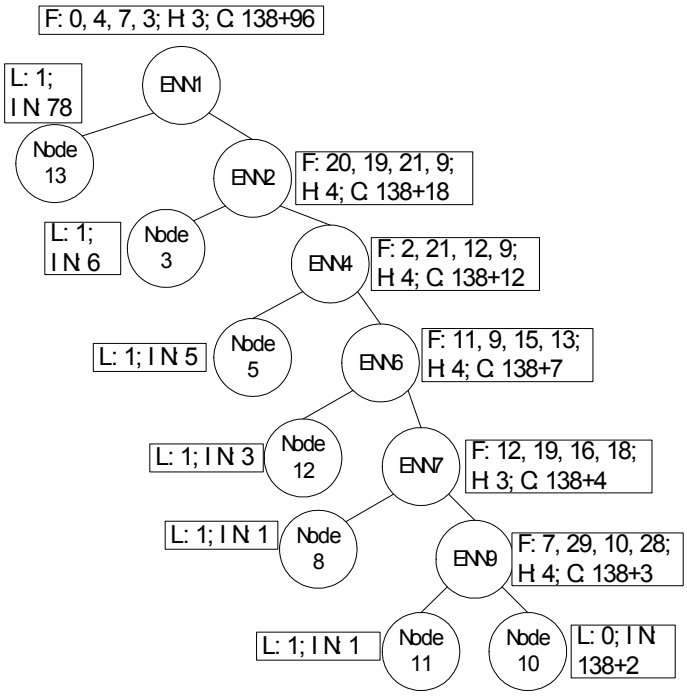


Figure 3. NNTree constructed through self-organized learning of features on Ionosphere data

Table 2. Comparison Data Result Of Dermatology

Methods	*Size*	N_H	R_{train}	R_{test}	*Time (Sec)*
MOO-GA	11.85	3.53	99.23%	95.71%	27.78
4 Points	12.05	3.48	99.18%	95.90%	22.30
8 Points	11.65	3.23	99.12%	96.25%	30.45

Table 3. Comparison Data Result Of E.coli

Methods	*Size*	N_H	R_{train}	R_{test}	*Time (Sec)*
MOO-GA	33.85	3.52	96.81%	80.47%	96.68
4 Points	42.00	3.84	93.23%	79.46%	124.00
8 Points	36.00	3.60	96.18%	81.63%	116.00

Table 4. Comparison Data Result Of Ionosphere

Methods	*Size*	N_H	R_{train}	R_{test}	*Time (Sec)*
MOO-GA	10.50	2.88	99.10%	90.75%	127.73
3Points	21.30	3.38	98.31%	87.76%	497.55
6 Points	13.45	3.23	98.84%	91.00%	337.50

Table 5. Comparison Data Result Of Liver

Methods	*Size*	N_H	R_{train}	R_{test}	*Time (Sec)*
MOO-GA	62.10	3.76	99.00%	60.43%	531.28
3 Points	59.25	3.78	72.49%	54.02%	377.45
6 Points	89.10	3.67	95.04%	60.59%	424.78

Thirdly, we come to evaluate time-consuming item. As shown in the four tables, time elapsed for each run was not increased much (even decreased) on most quantized datasets (Except Ionosphere dataset, most probably for its increased tree size). Namely, time computational complexity was almost not increased by construction NNTrees on quantized datasets.

5. Conclusions

An NNTree is actually a modular neural network with the overall structure being a DT, and each non-terminal node being an ENN. In this paper, we quantized the continuous inputs using self-orgainzed learning in each dimension, and constructed the NNTrees on the quantized datasets. Experiments seem to indicate that the NNTrees built from the quantized training data are equally good as those obtained from the original data. Further, the number of quantization points in each dimension is usually less than 10 for the databases we used. This means that 3 or 4 binary inputs are enough to represent each continuous input, and thus, the NNTrees so obtained are much more interpretable.

Acknowledgement: This work was supported in part by the Project of NSFC (Grant 60133010, 60272046, 60102011), National High Technology Project (Grant 2002AA143010) of China, and the Project of NSFJS (Grant BK2001042).

References

[1] Q. F. Zhao, "Evolutionary Design of Neural Network Tree -- Integration of Decision Tree, Neural Network and GA," *Proc. IEEE Congress on Evolutionary Computation (CEC'2001)*, Seoul, 2001, pp. 240-244.

[2] S. Mizuno and Q. F. Zhao, "Neural Network Trees with Nodes of Limited Inputs are Good for Learning and Understanding," *Proc. 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL2002),* Singapore, 2002,pp. 573-576.

[3] Taichirou Endou, Qiangfu Zhao, "Generation Of Comprehensible Decision Trees Through Evolution Of Training Data," *Proceedings of the 2002 Congress on Evolutionary Computation,* 12-17 May 2002, Vol. 2, Pages:1221 – 1225.

[4] Hiroshi tsukimoto, "Extracting Rules from trained neural networks," *IEEE Trans. Neural Networks.* Vol. 11, No.2, pp.377-389, 2000.

[5] Chun Lu, Qiangfu Zhao, Wenjiang Pei, and Zhenya He, "A Multiple Objective Optimization Based GA For Designing Interpretable And Comprehensible Neural Network Trees," *IEEE Int. Conf. Neural Networks & Signal Processing ,* China, 2003,Vol.1, pp. 518-521.

[6] Tom M. Mitchell, "Machine Learning," The McGraw-Hill Companies, Inc.,1997, pp. 52-79

[7] J.R.Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993

[8] C. A. C. Coello, C. "An Empirical Study of Evolution Techniques for Multiobjective Optimization in Engineering Design," *Ph.D.thesis*, Department of Computer Science, Tulane University, New Orleans, LA, 1996.

[9] T. Kohonen, "Self-organized formation of topologically correct feature maps," Biolog. Cybern.,Vol. 43, pp. 59-69, 1982.

[10] T. Kohonen, "The self-organizing map," Proc. IEEE, Vol. 78, No. 9, pp. 1464-1480, Sept. 1990.

[11] Zhang Yi, Pheng-Ann Heng, Ping-Fu Fung, "Winner-take-all discrete recurrent neural networks," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing,* Vol. 47 , Issue: 12 , pp. 1584 – 1589, Dec. 2000.

[12] Indiveri, G., Oswald, P., Kramer, J., "An adaptive visual tracking sensor with a hysteretic winner-take-all network," *IEEE International Symposium on Circuits and Systems,* 26-29 May, 2002. Vol. 2, pp. II-324 - II-327

[13] Raglin, A., Vorontsov, M., Chouikha, M., "Winner take all in a large array of opto-electronic feedback circuits for image processing," *2002 International Conference on Image Processin,* 22-25 Sept. 2002, Vol. 2, pp. II-349 - II-352

[14] Hung, Y.C., Liu, B.D., "An analog CMOS rank-order extractor with O(N) complexity using maximum/winner-take-all circuit," *2002 Asia-Pacific Conference on Circuits and Systems,* 28-31 Oct. 2002, Vol. 2, pp.389 – 394.

QinZhen Xu received the M.S. degree from Biomedical Engineering Department of Southeast University in 2000, and currently a Ph.D. student of Radio Engineering Department of Southeast University, China. Her research interest includes neural network tree, evolutionary algorithm and nonlinear signal processing.

Qinagfu Zhao received was an associate professor of this university from 1995 to 1999, associate professor of Tohoku University from 1993 to 1995, associate professor of Beijing Institute of Technology (BIT) from 1991 to 1993, and post-doctoral fellow at BIT from 1988 to 1991.Currently, he is a professor at the Multimedia Device Laboratory, University of Aizu.His research interests includes neural network tree, evolutionary algorithm and human face recognition.

Wenjiang Pei received M.S. degree and the Ph.D. degree from Nanjing University of Aeronautics and Astronautics in 1995 and 1997, respectively. Currently, He is a professor at Department of Radio Engineering, Southeast University, China. His research interest includes neural network, nonlinear dynamics, signal processing in networking, and secure communication.

Luxi Yang received the M.S. degree and Ph. D. degree in Electrical Engineering from Southeast University in 1990 and 1993, respectively. He is now a professor and the director of Digital Signal Processing Division in Radio Department of Southeast University. His major research interests include signal/image processing, blind signal processing, neural networks, and space-time signal processing for mobile communications.

Zhenya He is the Professor and Director of China key discipline of signal and information processing, Head of Neuro computing Science Research Center of Southeast University, and the National Presiding Scientist of China. Prof. He is Vice-President of China Neural Network Council and China Signal Processing Society, the Associate Director and Standing Editor of 5 National Journals of China, and Associate Editor of 2 International Journals of CSC, MSSP. Prof. He is Fellow of IEEE, CIC & CE, Member of TC on Blind Signal Processing of IEEE CAS Society, and Chair of IEEE CAS Shanghai Chapter, etc.