http://dx.doi.org/10.5392/IJoC.2013.9.4.011

Protein Disorder Prediction Using Multilayer Perceptrons

Sang-Hoon Oh
Department of Information Communication Engineering
Mokwon University, Daejon, 302-729, Korea

ABSTRACT

"Protein Folding Problem" is considered to be one of the "Great Challenges of Computer Science" and prediction of disordered protein is an important part of the protein folding problem. Machine learning models can predict the disordered structure of protein based on its characteristic of "learning from examples". Among many machine learning models, we investigate the possibility of multilayer perceptron (MLP) as the predictor of protein disorder. The investigation includes a single hidden layer MLP, multi hidden layer MLP and the hierarchical structure of MLP. Also, the target node cost function which deals with imbalanced data is used as training criteria of MLPs. Based on the investigation results, we insist that MLP should have deep architectures for performance improvement of protein disorder prediction.

***Key words**: Protein Disorder Prediction, Multilayer Perceptron, Error Function, Hierarchical Structure.*

1. INTRODUCTION

Proteins carry out many important functions indispensable for life and the study of protein structure is important for our understanding of many biological processes [1]. When a protein is in its functional state, it is called native. The native form of a protein is assumed to have a specific 3D structure and the loss of function is assumed to be associated with unfolding or loss of the specific 3D structure. Protein structure can be determined by the X-ray diffraction, NMR, homology alignment, and other methods [2]-[4].

The information flow from amino acid sequence to 3D structure is very important and this "protein folding problem" is considered to be one of the "Great Challenges of Computer Science" [5], [6]. The protein folding problem includes the prediction of order and disorder. A protein region is defined as disordered if it is devoid of stable secondary structure [7], [8]. Recognition of disordered regions in a protein is important for two reasons: reducing bias in sequence similarity analysis by avoiding alignment of disordered regions against ordered ones, and helping to delineate boundaries of protein domains to guide structural and functional studies [7]. Accurate recognition of disordered regions can be applied to enzyme specificity studies, function recognition, and drug design [4]. However, there are several categories of disorder such as molten globules, partially unstructured proteins, and random coils [7]. And no commonly agreed definition of protein disorder exists [2].

Intrinsically disordered proteins generally have a biased amino acid composition [7]. G, S, and P are disorder-promoting amino acids. W, F, I, Y, V and L are order-promoting amino acids, while H and T are considered neutral with respect to disorder. However, using sequence composition as the sole predictive parameter of disorder is not reliable [7].

Disordered regions can be indirectly predicted by experimental methods such as X-ray crystallography, NMR-, Raman-, CD-spectroscopy, and hydrodynamic measurements [2]. Each of these methods detects different aspects of disorder resulting in several operational definitions of protein disorder.

Alternatively, machine learning approaches to determine whether disordered regions are common have been proposed. Romeo et al. proposed PONDR method, which constructed feature extraction data through p-feature selection and PCA (principal component analysis) and then trained MLP (multi-layer perceptron) using EBP (error back-propagation) algorithm [1]. In the PONDR method, general predictors are trained using all available disordered examples. Family-specific predictors are trained to predict a particular type of disorder. Also, hybrid predictor combines family-specific predictors into more general disorder predicting systems by using an arbiter neural network decision when the base predictors disagree. However, there is severe imbalance between disordered and ordered regions. The PONDR method used an artificial procedure to make the data balanced.

Yang and Thomson proposed BBFNN (bio-basis function neural network) which resembles GPFN (Gaussian potential function network) [3]. In the method, bio-basis function was designed based on homology alignment score and the weights of the final layer were calculated with pseudo-inverse method. They also proposed RONN in order to handle the variable length of disordered/ordered regions [4]. RONN has weakness particularly in the detection of short regions of disorder and in defining the first and last residues of disordered regions.

* *Corresponding author, Email: shoh@mokwon.ac.kr*
Manuscript received Aug. 16, 2013; revised Nov. 27, 2013; accepted Dec. 06, 2013

Linding et al. proposed DisEMBL which consisted of three neural networks, of which each one detects a separately defined disordered regions such as loops/coils, hot loops, and missing coordinates in X-ray structure [2]. Possibly because of the small number of positive (disordered region) samples, Linding et al. insisted that networks with many hidden nodes performed no better than those with few. So, they used only five hidden nodes but did not consider the imbalance of data to train neural networks [2].

Data imbalance is reported in a wide range of applications such as bio-medical diagnoses [9], gene ontology [10], remote sensing [11], credit assessment [12], etc. Classifiers developed under the assumption of balanced class priors show poor performance for the imbalanced data problems including the protein disorder prediction.

When dealing with the prediction of protein disorder problem, in this paper, we considered the imbalance of data to train MLPs. Also, we investigate structural possibility of MLP for the protein disorder prediction problem. In section 2, we briefly introduce the EBP algorithm of MLP and the target node method to deal with the imbalanced data in the EBP scheme. In section 3, we propose many architectures of MLP for the protein disorder prediction problem and show simulation results. Finally, section 4 concludes this paper.

2. ERROR BACK-PROPAGATION ALGORIHTM AND IMBALANCED DATA

Among many supervised learning models in the machine learning field, we select MLP as a predictor of disordered proteins because of its arbitrary function approximation capability [13].

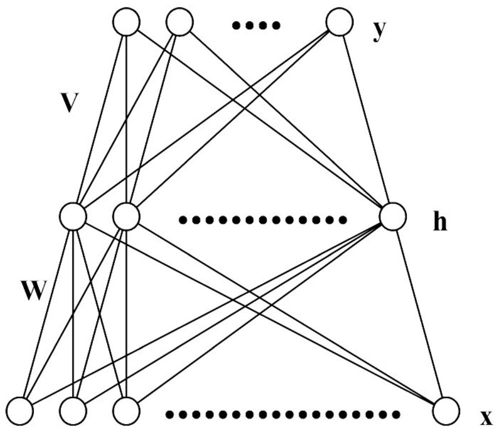

Fig. 1. The architecture of a multilayer perceptron.

Consider an MLP consisting of N inputs, H hidden nodes, and M output nodes, which is denoted as an "N-H-M MLP". When a sample $\mathbf{x}^{(p)} = [x_1^{(p)}, x_2^{(p)}, \ldots, x_N^{(p)}]$ $(p = 1,2,\ldots,P)$ is presented to the MLP, by the forward propagation, the j-th hidden node is given by

$$\begin{aligned} h_j^{(p)} &= h_j(\mathbf{x}^{(p)}) \\ &= \tanh((w_{j0} + \sum_{i=1}^{N} w_{ji} x_i^{(p)})/2), \ j = 1,2,\ldots,H. \end{aligned} \tag{1}$$

Here, w_{ji} denotes the weight connecting x_i to h_j and w_{j0} is a bias. The k-th output node is

$$y_k^{(p)} = y_k(\mathbf{x}^{(p)}) = \tanh(\hat{y}_k^{(p)}/2), \ k = 1,2,\ldots,M, \tag{2}$$

where

$$\hat{y}_k^{(p)} = v_{k0} + \sum_{j=1}^{H} v_{kj} h_j^{(p)} . \tag{3}$$

Also, v_{k0} is a bias and v_{kj} denotes the weight connecting h_j to y_k.

Let the desired output vector corresponding to the training sample $\mathbf{x}^{(p)}$ be $\mathbf{t}^{(p)} = [t_1^{(p)}, t_2^{(p)}, \ldots, t_M^{(p)}]$, which is coded as follows:

$$t_k^{(p)} = \begin{cases} +1 & \text{if } \mathbf{x}^{(p)} \text{ originates from class } k \\ -1 & \text{otherwise.} \end{cases} \tag{4}$$

As a distance measure between the actual and desired outputs, we usually use the squared error function for P training samples defined by

$$E = \sum_{p=1}^{P} \sum_{k=1}^{M} \left(t_k^{(p)} - y_k^{(p)}\right)^2 . \tag{5}$$

To minimize E, weights v_{kj} 's are iteratively updated by

$$\Delta v_{kj} = -\eta \frac{\partial E}{\partial v_{kj}} = \eta \delta_k^{(p)} h_j^{(p)}, \tag{6}$$

where

$$\delta_k^{(p)} = -\frac{\partial E}{\partial \hat{y}_k^{(p)}} = \left(t_k^{(p)} - y_k^{(p)}\right) \frac{\left(1 - y_k^{(p)}\right)\left(1 + y_k^{(p)}\right)}{2} \tag{7}$$

is the error signal and η is the learning rate. Also, by the backward propagation of the error signal, weights w_{ji} 's are updated by

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} = \eta x_i^{(p)} \sum_{k=1}^{M} v_{kj} \delta_k^{(p)} . \tag{8}$$

The above weight-updating procedure is the EBP algorithm [14], which does not consider any imbalance among classes.

In the protein disorder prediction problem, the positive (disordered region) samples are much less than the negative (ordered region) samples. This imbalance severely degrades the performance of protein disorder prediction. To resolve the imbalance, Romeo et al. adopted an artificial procedure to make the data balanced [1]. Linding et al. reported that MLP with few hidden nodes is better than MLP with many hidden nodes [2]. This strategy degrades the approximation capability of MLPs [13]. Contrary to these artificial or heuristic methods, the better way is to use an algorithmic approach which was proposed to strengthen learning with regards to the positive samples [16].

Consider two-class problems with imbalanced data sets [15]. Assume that one is the minority class C_1 with P_1 training samples and the other is the majority class C_2 with P_2 training samples ($P_1 << P_2$). If we use the conventional EBP algorithm to train the MLP, weight-updating is overwhelmed by the majority class samples and this severely distorts the boundary between the two classes [16]. This causes poor classification performance for the minority class even though samples in the minority class have a high misclassification cost [16].

In order to prevent the boundary distortion, the target node method was proposed whose error function was defined by

$$E_{TN} = -\sum_{p=1}^{P}\left[\int \frac{t_1^{(p)n+1}\left(t_1^{(p)} - y_1^{(p)}\right)^n}{2^{n-2}\left(1 - y_1^{(p)2}\right)} dy_1^{(p)} + \int \frac{t_2^{(p)m+1}\left(t_2^{(p)} - y_2^{(p)}\right)^m}{2^{m-2}\left(1 - y_2^{(p)2}\right)} dy_2^{(p)}\right], \quad (9)$$

where *n* and *m* (*n*<*m*) are positive integers and $t_k^{(p)}$ is coded as in (4) [16]. If *n*=*m*, E_{TN} is the same as the nth order extension of the cross-entropy error function [17]. Then, the error signal of the output layer is given by

$$\delta_k^{(p)} = -\frac{\partial E_{prop}}{\partial \hat{y}_k^{(p)}} = \begin{cases} t_k^{(p)n+1}\left(t_k^{(p)} - y_k^{(p)}\right)^n / 2^{n-1} & \text{for } k = 1, \\ t_k^{(p)m+1}\left(t_k^{(p)} - y_k^{(p)}\right)^m / 2^{m-1} & \text{for } k = 2. \end{cases} \quad (10)$$

The parameters *n* and *m* controls the updating amount of weights whether the target nodes are for the minority or majority classes. Since *n*<*m* and $-1 \le y_k^{(p)} \le 1$, the error signal for the target node of minority class is greater than or equal to the error signal for the target node of majority class. This effect can prevent the boundary distortion problem [16], [18].

Also, in order to fix the imbalance of targets for '1' and '-1', $\delta_k^{(p)}$'s are regulated as $\gamma\delta_k^{(p)}$ with the parameter $\gamma = P_1 / P_2$ in the case that (*k*=1 and $t_k^{(p)} = -1$) or (*k*=2 and $t_k^{(p)} = 1$) [16]. Then, the associated weights are updated in proportion to the error signals, which is the same procedure as in the EBP algorithm [14].

3. MLPs FOR PROTEIN DISORDER PREDICTION

In this section, we train MLPs to be a protein disorder predictor. The learning algorithm is the target node method which shows better performance in imbalanced data problems [16]. Still there are many possibilities in the architecture of MLPs and we will try to find a better architecture for the protein disorder prediction.

The protein disorder prediction database was supplied from KIAS(Korea Institute for Advanced Study). A total of 215,612 feature vectors were extracted from 723 proteins with 15 window size. Each feature vector consists of 330 dimensional sequence profile, 45 dimensional secondary structure profile, 16 dimensional solvent accessibility profile, and 17 dimensional hydrophobicity profile. So, the feature vector is totally 408 dimensional.

Firstly, we simulated the protein disorder prediction with a single hidden layer MLP of 408 inputs, 20 hidden nodes, and 2 output nodes. Since the protein disorder prediction is imbalanced, we used the target node method given by Eq. (9) with *n=2* and *m=8* to train MLPs for 5000 epochs. Nine times simulations were conducted with initializations of MLP weights uniformly on $\left[-1\times10^{-4}, 1\times10^{-4}\right]$. This initialization range of MLP weights is to avoid premature saturation phenomenon of learning [19]. In each simulation, we performed five-fold (1-out of-5) cross-validation for performance evaluation. When data is imbalanced, the total accuracy heavily depends on the accuracy of majority class and the total accuracy is not adequate as a performance measure. Accordingly, as performance criteria, we used the accuracy of minority (disordered region) class and the geometric mean of majority (ordered region) class accuracy and minority class accuracy [9]. The forty five simulation results consisted of nine times MLP initialization and five times cross-validation are averaged and the best performance during 5000 training epochs is in the Table 1. The accuracy of disordered region class and the geometric mean for training samples are 91% and 89.4%, respectively. For validation samples, the accuracy of disordered region class and the geometric mean are 79.03% and 80.46%, respectively.

Table 1. The simulation results for the 408-20-2 MLP. 408 is the number of input nodes, 20 is the number of hidden nodes, and 2 is the number of output nodes. G-Mean denotes the geometric mean

Training Samples		Validation Samples	
Accuracy (Disorder Class)	G-Mean of Two Classes	Accuracy (Disorder Class)	G-Mean of Two Classes
91.00%	89.40%	79.03%	80.46%

Alternatively, we tried a hierarchical architecture of MLPs. Since each input vector consists of four profiles, we allocate $MLP_i (i = 1,2,3,4)$ for each profile in the input vector. That is, MLP_1 is for the 330 dimensional sequence profile, MLP_2 is for the 45 dimensional secondary structure profile, MLP_3 is for the 16 dimensional solvent accessibility profile, and MLP_4 is for the 17 dimensional hydrophobicity profile. All these $MLP_i (i = 1,2,3,4)$ have 20 hidden nodes and two output nodes, respectively. The totally eight output node values from $MLP_i (i = 1,2,3,4)$ are presented to the judge MLP, which integrates the classification information of $MLP_i (i = 1,2,3,4)$ and makes a final decision. The architecture of judge MLP is 8-20-2. The initialization and training methods are the same with the single hidden layer MLP. The performance is evaluated after averaging of forty five simulation results and shown in Table 2.

Since each profile has different characteristics, $MLP_i (i = 1,2,3,4)$ show different performances. Among them, MLP_1 and MLP_3 are better and MLP_4 is the worst. This means that the hydrophobicity profile is more complex than the other profiles. After integration of information from $MLP_i (i = 1,2,3,4)$, the judge MLP improves the performance. However, the performance of judge MLP for validation samples is slightly inferior to that of single hidden layer MLP.

Table 2. The simulation results for the hierarchical MLPs. The 330-20-2 MLP_1 is for the sequence profile, the 45-20-2 MLP_2 is for the secondary structure profile, the 16-20-2 MLP_3 is for the solvent profile, and the 17-20-2 MLP_4 is for the hydrophobicity profile. The 8-20-2 judge MLP is for the final decision.

	Training Samples		Validation Samples	
	Accuracy (Disorder Class)	G-Mean of Two Classes	Accuracy (Disorder Class)	G-Mean of Two Classes
MLP_1	85.86%	85.45%	75.20%	78.63%
MLP_2	74.66%	76.64%	73.04%	75.72%
MLP_3	77.12%	79.31%	77.71%	79.08%
MLP_4	66.35%	70.02%	66.51%	68.86%
Judge MLP	90.27%	86.59%	78.40%	80.37%

The performance of MLP depends on the number of hidden nodes as well as the number of hidden layers. Contrary to the first and second simulations which used MLPs with a single hidden layer, we tried to increase the number of hidden layers from one to three. Here, "$N - H_1 - H_2 - H_3 - M$ MLP" denotes MLP with H_1, H_2, and H_3 nodes in the first, second, and third hidden layer, respectively.

Now, we tried the architecture of MLPs such as 408-2-2-2-2, 408-4-4-4-2, and 408-20-20-20-2. The initialization and training methods are the same with the first simulation. Also, the performances are evaluated using the averages of forty five simulation results and shown in Table 3. Comparing Table 3 with Table 1, we can find that the three hidden layer MLP with 20 hidden nodes in each hidden layer attains better performance for training samples and similar performance for the validation samples. This is due to the specialization to the training samples with increased hidden nodes. That is, the (c) case in Table 3 has $20 \times 3 = 60$ hidden nodes.

As a final trial, we simulated the hierarchical architecture of MLPs with three hidden layers. Here, each $MLP_i (i = 1,2,3,4)$ has three hidden layers and the judge MLP also has three hidden layers. In each hidden layer of $MLP_i (i = 1,2,3,4)$ and the judge MLP, we used 20 hidden nodes. The initialization and training methods are the same with the first simulation. As in the previous simulations, the performances evaluated using the average of the forty five simulation results are in Table 4.

Table 3. The simulation results with three hidden layer MLPs, whose architectures are (a) 408-2-2-2-2 (two nodes in each hidden layer), (b) 408-4-4-4-2 (four nodes in each hidden layer), and (c) 408-20-20-20-2 (twenty nodes in each hidden layer).

	Training Samples		Validation Samples	
	Accuracy (Disorder Class)	G-Mean of Two Classes	Accuracy (Disorder Class)	G-Mean of Two Classes
(a)	80.26%	81.11%	80.56%	80.18%
(b)	82.30%	81.67%	79.65%	80.52%
(c)	96.98%	89.71%	80.67%	80.12%

Comparing Tables 2 and 4, the performance of judge MLPs is similar. However, by increasing the number of hidden layer, the Accuracy of Disorder Class of MLP_4 was improved very much. Also, the (c) case in Table 3 shows a similar tendency. Thus, there are possibilities of performance improvement with increasing the number of hidden layers. Although we can improve the performance with increased hidden nodes, this causes specialization of learning to training samples and finally degradation of performance for test samples. Therefore, we pursue to increase the number of hidden layers. This argument coincides with the interest increasing of neural network community in deep belief networks [20], [21].

Table 4. The simulation results for the hierarchical MLPs with three hidden layers. MLP_1 is 330-20-20-20-2, MLP_2 is 45-20-20-20-2, MLP_3 is 16-20-20-20-2, MLP_4 is 17-20-20-20-2, and the judge MLP for the final decision is 8-20-20-20-2.

	Training Samples		Validation Samples	
	Accuracy (Disorder Class)	G-Mean of Two Classes	Accuracy (Disorder Class)	G-Mean of Two Classes
MLP_1	87.34%	84.51%	74.58%	78.46%
MLP_2	75.15%	76.60%	73.62%	75.89%
MLP_3	76.42%	79.27%	79.62%	79.04%
MLP_4	84.12%	70.95%	84.06%	68.75%
Judge MLP	90.60%	86.38%	78.46%	80.36%

The poor performance of MLP is due to the specialization to training samples, or in some cases, MLPs cannot fit the true-function described by the training samples. As a new strategy to resolve these problems, the deep architecture of MLP has been proposed [21]. Since the deep belief network has many hidden layers, it is very difficult to successfully train the deep network. As an initialization methodology for successful training, the RBM (Restricted Boltzmann Machine) had been proposed [20]. We tried various architectures of MLPs and the fruit we attained is that increasing the number of hidden layers can improve the performance. So, we will adopt the deep architecture of MLPs initialized with the RBM as a next challenging method to the protein disorder prediction.

5. CONCLUSIONS

In this paper, we investigated the possibilities of MLP as a machine learning methodology for protein disorder prediction. The single hidden layer MLP, hierarchical MLPs, three hidden layer MLP, and hierarchical MLPs with three hidden layers are simulated. Contrary to the others, we trained MLPs with the target node method which can deal with imbalanced data problems. Since the protein disorder prediction is heavily imbalanced, MLPs must be trained with the learning algorithm which is developed for the imbalanced data.

With the simulation results, it was very difficult to improve the performance for the protein disorder prediction problem. Anyway, there was a possibility that increasing the number of hidden layers can improve the performance of protein disorder prediction. This argument coincides with the high interests in the deep architecture field of neural network community. As a next step, we will try the deep belief network with initialization using RBM for performance improvement of protein disorder prediction.

REFERENCES

[1] P. Romero, Z. Obradovic, and A. K. Dunker, "Intelligent data analysis for protein disorder prediction," Artificial Intelligence Review, vol. 14, 2000, pp. 447-484.

[2] R. Linding, L. J. Jensen, F. Diella, P. Bork, T. J. Gibson, and R. B. Russell, "Protein disorder prediction: Implications for structural proteomics," Structure, vol. 11, 2003, pp. 1453-1459.

[3] Z. R. Yang and R. Thomson, "Bio-basis function neural network for prediction of protease cleavage sites in proteins," IEEE Trans. Neural Networks, vol. 16, 2005, pp. 263-274.

[4] Z. R. Yang, R. Thomson, P. McNeil, and R. M. Esnouf, "RONN: the bio-basis function neural network technique applied to the detection of natively disordered regions in proteins," Bioinformatics, vol. 21, 2005, pp. 3369-3376.

[5] FCCST, Grand Challenges 1993: High performance computing and communications, A report by the committee on physical, mathematical, and engineering sciences, Federal coordinating council for science and technology.

[6] O. Noivirt-Brik, J. Prilusky, and J. L. Sussman, "Assessment of disorder predictions in CASP8," Proteins, vol. 77, 2009, pp. 210-216.

[7] F. Ferron, S. Longhi, B. Canard, and D. Karlin, "A practical overview of protein disorder prediction methods," PROTEINS: Structure, Function, and Bioinformatics, vol. 65, 2006, pp. 1-14.

[8] B. He, K. Wang, Y. Liu, B. Xue, V. N. Uversky, and A. K. Dunker, "Predicting intrinsic disorder in proteins: an overview," Cell Research, vol. 19, 2009, pp. 929-949.

[9] P. Kang and S. Cho, "EUS SVMs: ensemble of under-sampled SVMs for data imbalance problem," Proc. ICONIP'06, 2006, pp. 837-846.

[10] R. Bi, Y. Zhou, F. Lu, and W. Wang, "Predicting gene ontology functions based on support vector machines and statistical significance estimation," Neurocomputing, vol. 70, 2007, pp. 718-725.

[11] L. Bruzzone, and S. B. Serpico, "Classification of Remote-Sensing Data by Neural Networks," Pattern Recognition Letters, vol. 18, 1997, pp. 1323-1328.

[12] Y. M. Huang, C. M. Hung, and H. C. Jiau, "Evaluation of Neural Networks and Data Mining Methods on a Credit Assessment Task for Class Imbalance Problem," Nonlinear Analysis, vol. 7, 2006, pp. 720-747.

[13] K. Hornik, M. Stincombe, and H. White, "Multilayer feed-forward networks are universal approximators," Neural Networks, vol. 2, 1989, pp. 359-366.

[14] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing*, Cambridge, MA, 1986.

[15] N. V. Chawla, K. W. Bowyer, L. O. all, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," J. Artificial Intelligence Research, vol. 16, 2002, pp. 321-357.

[16] S. H. Oh, "Error back-propagation algorithm for classification of imbalanced data", Neurocomputing, vol. 74, 2011, pp. 1058-1061.

[17] S. H. Oh, "Improving the Error Back-Propagation Algorithm with a Modified Error Function," IEEE Trans. Neural Networks, vol. 8, 1997, pp. 799-803.

[18] S. H. Oh, "A Statistical Perspective of Neural Networks for Imbalanced Data Problems," Int. Journal of Contents, vol. 7, no. 3, 2011, pp. 1-5.

[19] Y. Lee, S. H. Oh, and M. W. Kim, "An Analysis of Premature Saturation in Back-Propagation Learning," Neural Networks, vol. 6, 1993, pp. 719-728.

[20] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," Science, vol. 313, 2006, pp. 504-507.

[21] Y. Bengio, "Learning Deep Architecture for AI," Foundations and Trends in Machine Learning, vol. 2, 2009, pp. 1-127.

Sang-Hoon Oh
He received his B.S. and M.S degrees in Electronics Engineering from Pusan National University in 1986 and 1988, respectively. He received his Ph.D. degree in Electrical Engineering from Korea Advanced Institute of Science and Technology in 1999. From 1988 to 1989, he worked for LG semiconductor, Ltd., Korea. From 1990 to 1998, he was a senior research staff in Electronics and Telecommunication Research Institute (ETRI), Korea. From 1999 to 2000, he was with Brain Science Research Center, KAIST. In 2000, he was with Brain Science Institute, RIKEN, Japan, as a research scientist. In 2001, he was an R&D manager of Extell Technology Corporation, Korea. Since 2002, he has been with the Department of Information Communication Engineering, Mokwon University, Daejon, Korea, and is now a full professor. Also, he was with the Division of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, USA, as a visiting scholar from August 2008 to August 2009. His research interests are machine learning, speech signal processing, pattern recognition, and bioinformatics.