연결자 제거를 위한 간단한 알고리즘과 모의 랜덤 신호 분류에의 응용

원 용 관† · 민 병 의†

요 약

본 논문은 신경망에서 불필요한 연결자(weights and biases)를 제거하기 위한 일반적인 back-propagation 알고리즘의 간단한 변형을 소개한다. 이는 연결자들의 절대치 크기의 분포, 분할 경계선의 분석 및 sigmoid 함수의 비선형성에 기초하여 개발되었다. 신경망의 초기 학습 후, 이 알고리즘은 그 절대치의 크기가 임계치(threshold) 보다 적은 모든 연결자들의 값을 영으로 할당 함으로서 이들을 제거하게 된다. 그런후, 중요한 연결자들의 복구를 위해 모든 연결자들을 포함하여 반복 학습을 실시한다. 이진함수들을 이용한 학습에서, 이 알고리즘은 이론적 최소 구조를 실현하였고, 함수를 푸는데 불필요한 연결자들을 제거하였다. 모의 랜덤 신호 분류에의 응용에 있어서, 본 알고리즘으로부터 얻어낸 결과는 쉬운 문제가 간단한 신경망을 필요로 하며 낮은 오분류율을 발생한다는 일반적인 개념과 일치하였다. 또한, 본 제안된 알고리즘은 overfitting과 형태소(pattern) 암기의 문제점들을 줄임으로서 기존의 알고리즘보다 더 좋은 결과를 보여주었다.

A Simple Connection Pruning Algorithm and its Application to Simulated Random Signal Classification

Yonggwan Won† · Byungeui Min†

ABSTRACT

A simple modification of the standard back-propagation algorithm to eliminate redundant connections (weights and biases) is described. It was motivated by speculations from the distribution of the magnitudes of the weights and the biases, analysis of the classification boundary, and the nonlinearity of the sigmoid function. After initial training, this algorithm eliminates all connections of which magnitude is below a threshold by setting them to zero. The algorithm then conducts retraining in which all weights and biases are adjusted to allow important ones to recover. In studies with Boolean functions, the algorithm reconstructed the theoretical minimum architecture and eliminated the connections which are not necessary to solve the functions. For simulated random signal classification problems, the algorithm produced the results which is consistent with the idea that easier problems require simpler networks and yield lower misclassification rates. Furthermore, in comparison, our algorithm produced better generalization than the standard algorithm by reducing overfitting and pattern memorization problems.

† 정 회 원 : 한국 전자 통신 연구소, 인공지능 연구실
논문접수 : 1995년 12월 8일, 심사완료 : 1996년 1월 31일

1. INTRODUCTION

Among the factors that affect the generalization, the architecture of the network is the most important one because its relationship to the number of training examples and problem difficulty mainly affects the generalization capability [1]. As a rule of thumb, the smallest network that fits the data leads to good generalization. Some previous researchers used learning theory to estimate the appropriate size of a network [2, 3, 4]. However, choosing an appropriate network architecture is still not obvious. On the other hand, large initial networks converge faster and are less probable to be trapped in local minima [1, 5]. Therefore, it is reasonable to train a large initial network and prune the excess architecture without degrading the performance.

Many of previous investigations have been concerned with obtaining an optimal architecture by trimming fat from a larger initial network. They are categorized into two classes: one is unit level optimization [6, 7, 8, 9, 10, 11, 12, 13, 14], and the other is connection level optimization [15, 16, 17, 18]. Connection pruning completely eliminates weights and biases that are deemed to be redundant. In general, previous approaches used two methods: sensitivity estimation and penalty terms in the error function, which both require extra computational complexity. Extensive review of pruning algorithms is available from [5].

This paper is composed of six sections. We describe the motivation for developing the algorithm in Section 2. It is followed by a section which describes a simple repetitive two-step training algorithm that prunes redundant connections for multi-layer feedforward networks. In Section 4, we justify the algorithms by showing that the algorithm constructed networks that had the theoretical minimum architecture and reflected the logical rules of XOR and Rule-and-Exception problems. Results of simulated random signal classification studies are described in Section 5. It includes a set simulations which verify that our pruning algorithm helps better generalization. Finally, conclusions and possible future works are described in Section 6.

2. MOTIVATIONS FOR THE ALGORITHM

To develop an algorithm for pruning the redundant connections, we examined the weight distributions of different architectures, effects of pruning connections on the classification boundaries, and the nonlinearity of the neural network units.

2.1. Magnitude Distribution of the Connections

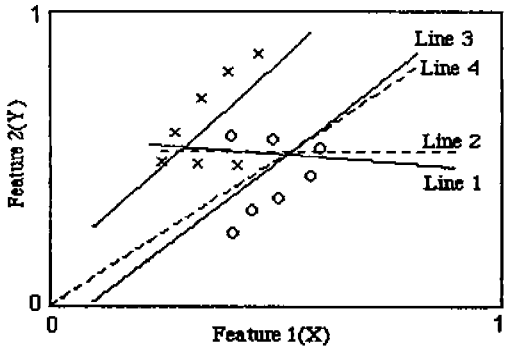
The final weight set of a trained neural network is totally dependent on the initial weights, training conditions, and the initial architecture. We investigated the effects of different architectures on the final weight sets. Theoretically, the XOR network has a minimum architecture that has a single hidden layer with two units [1]. For the XOR problem, the number of connections in small magnitude range increased as the number of the hidden units increased, while those in large magnitude range remained similar [19]. This finding suggests that weights of small magnitude are redundant.

2.2. Boundary Analysis and Pruning Effects

Multi-layer feed-forward networks draw the different boundaries depending upon the number of the layers and the number of the units in each layer [20]. The shape of the boundaries is different depending upon the weight and the bias values. In other words, the solution for weight value defining a particular boundary is not unique. Thus, a small tilt of the decision boundary, which is equivalent to a small change of the weight values, does not change the total performance of the trained network, and it may improve the performance sometimes.

Functional analysis and effects of pruning on the boundaries are well described in [19]. We show an example which can provide a speculation. Figure 1

illustrates the scatter diagram for the patterns with two features for two classes, which are separated by the three solid boundary lines, each defined by a unit with two weights and a bias. The two diagonal line have two weights which have similar magnitude, while the near horizontal line, designated as "Line 1", has one large magnitude weight and one small magnitude. In this situation, pruning a small magnitude weight converts the near horizontal line "Line 1" to the horizontal line, designated as "Line 2". Furthermore, pruning a small magnitude bias converts "Line 3" to another diagonal line passing through the origin, designated as "Line 4". However, new boundaries produce no change in the separation of the classes.



(Figure 1) Scatter diagram for the pattern sets with two features for two classes. Two classes can be separated by three boundary lines

2.3. Characteristics of the Sigmoidal unit

Training a network to discriminate the classes forces the output of the most units in a trained network to approach one of the extreme values of the non-linear sigmoidal mapping function, i.e., 1 or 0. In this situation, a small change of the netinput can be ignored in non-linear mapping to its output. The following mathematical description reflects this approximation on the netinput. The netinput to each unit can be separated into two summations as

$$\begin{aligned} \mathrm{Net}_{pj} &= W_j \cdot O_p \\ &= \sum_i W_{ji(small)}\, O_{pi} + \sum_i W_{ji(large)}\, O_{pi} \\ &= \sum_i W_{ji(small)}\, O_{pi} + \mathrm{Net}'_{pj} \end{aligned} \tag{1}$$

where p indicates the pattern index, W_j and O_p are the weight vector for unit j and corresponding input vector, respectively. In this equation, $W_{ji(small)}$ is a small magnitude weight, $W_{ji(large)}$ is a large magnitude and weight. The first term contributes relatively little to the netinput, and the second term contributes most of the netinput. Therefore, without any critical change on the output, Equation 1 can be approximated by

$$\begin{aligned} \mathrm{Net}_{pj} &\cong \mathrm{Net}'_{pj} \\ &= \sum_i W^{pj}_{ji(large)}\, O_{pi} \end{aligned} \tag{2}$$

2.4. Discussion

The speculations described in the previous three subsections allow us to simply prune the weights which have small magnitude. However, the distortion to the trained network generally increases the error, and there may be some situations where the small weights have important effects. In other words, the small tilt of the boundary can be critical in some classifications. Therefore, two questions should be considered when designing the connection pruning algorithm: how to compensate for the increased error and how to recover the critical tilt of the boundary.

In back-propagation training, in general, the magnitudes of the weights grow and the error decreases as the number of training epoch increases [19]. Also the weight change is bigger with bigger errors. Based upon these properties of the learning scheme, simply more training of the pruned network is most likely to compensate for the increased error. More training also can recover the critical tilt of the boundary. When a boundary can not be tilted because the small tilt is very critical, the small tilt of that boundary causes very big increase of the error, and this big error results in a big change on the weights

during the next training step. Finally, the weights which define the boundary grow fast and the boundary will recover in the same shape as before, however, with larger magnitude weights.

3. CONNECTION PRUNING ALGORITHM

This section describes the simple connection pruning algorithm which was developed based on the motivations and discussions in the previous section. The pruning algorithm first trains the network using the standard back-propagation learning scheme to adjust all weights (initial back-propagation training). When this initial training reaches either the specified error or number of maximum training epochs, all weights and biases with a magnitude below a specified threshold value are set to zero (i.e., pruning). If this pruning does not increase the error, the algorithm terminates; otherwise it resumes training (i.e., retraining) by again adjusting all the connections, also using the back-propagation learning scheme, until the network reaches either the error obtained by the initial back-propagation training or the specified number of epochs for each retraining cycle. This two-step process, retraining and pruning, continues until the error caused by pruning is less than that obtained by the initially trained network or until the total number of retraining epochs reaches its predefined value. The following pseudo code summarizes this pruning algorithm:

Initialize the Network.
DO update the weights by back-propagation learning.
UNTIL the number of epoch reaches a specified value OR the error become less than a specified value.
Prune the weights.
WHILE the total training epoch does not reach a specified number AND the error with pruned network is larger than the error at the end of the initial back-propagation learning.
DO update the weights by back-propagation learning.
UNTIL the specified number of retraining epochs per retraining cycle is reached OR the error becomes less than or equal to the error at the end of the initial back-propagation learning.
Prune the weights.
ENDWHILE

4. Experiments with XOR and Rule-and-Exception Networks.

This section presents some experimental results obtained from learning Boolean functions: XOR and Rule-and-Exception. These problems are good examples to justify that our algorithm constructs networks that have a minimum architecture and preserve the logical rule of the functions.

4.1. XOR Networks

In the study of the XOR problem, we used a network having a single hidden layer which is designated by 2: n: 1, where n ranged from two to twenty. Under these conditions, an efficient solution to the XOR problem contains two hidden units with six weights and three biases [1]. We trained each network with our pruning algorithm. The threshold value for each pruning was selected to be the closest integer value to the ninth largest magnitude value.

〈Table 1〉 Complexity of XOR network.

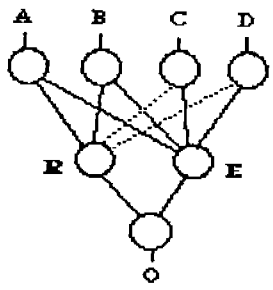
No. of Hidden units	3	4	6	8	10	15	20
Initial Network	9(4)	12(5)	18(7)	24(9)	30(11)	45(16)	60(21)
Pruned Network	6(2)	8(1)	9(1)	7(1)	8(1)	7(1)	11(0)

Table 1 shows the number of the weights and biases in the initial networks (second row) and those

in the network trained with our pruning algorithm (third row). For each network, the first number represents the number of weights and the second number in parentheses represents the number of biases. These data indicate that the algorithm always pruned the networks to one that contained eight to eleven weights and biases, which compares favorably to the ideal value of nine. Furthermore, in most cases, further analysis of the final networks [16] indicated that this pruning algorithm can reconstruct the most efficient two hidden unit network.

4.2. Rule-and-Exception Network

In this problem, the network learns the Boolean function AB + A'B'C'D' [16]. The output unit should be "on" whenever both unit A and B are "on", which is called "Rule", or when all the input units are "off", which is called "Exception". With a two hidden unit network shown in Figure 2, back-propagation learning reaches a solution with one hidden unit (R) responding to the "Rule" and the other (E) responding to the "Exception". Clearly, the "Rule" is more important to the solution than the "Exception" since the "Rule" accounts for 15 out of 16 patterns. Therefore, the trained network should show this aspect and helps prune the network. In other words, since the unit R accounts for all the patterns except the pattern 0000 and the unit E accounts for the pattern 0000, the connection W_{RC} and W_{RC} can be pruned without any misclassification, and the training algorithm should show this phenomenon.



(Figure 2) Network for the Rule-and-Exception problem. One hidden unit (R) responds to the "Rule" and the other (E) responds to the "Exception".

With this Rule-and-Exception problem, we examined whether pruning any weight changed the classification performance. In our experiments, the connections WRC and WRD had the smallest strengths, as in Karnin's study [16], and pruning them did not change the performance of the original network in classifying all 16 patterns. However, pruning only WEC and WED which are two least sensitive weights in Karnin's study caused the misclassification of the pattern 0000. Pruning all the connections to the unit E also caused the misclassification of the pattern 0000, which is also the same result as Karnin's. In summary, this result also justifies that our pruning method eliminates the redundant connections.

5. SIMULATED RANDOM SIGNAL CLASSIFICATION

This section presents the results from comparing the generalization performance of the standard back-propagation (unpruned) and this pruning algorithm for the simulated random signal classification problems. It also describes the pruning performance of our algorithm.

5.1. RANDOM SIGNAL GENERATION PROCESS

The random signal generation process [21] produces a set of class patterns using a mathematical model which generates pulse patterns with exponentially damped oscillatory edges. This model has five parameters: amplitude, starting time, pulse width, exponential coefficient and frequency of oscillations. For each pattern, the process selected the values for those parameters from a Gaussian distribution with a specified mean and variance. In addition, the process perturbed each pattern with random values specified by signal-to-noise ratio to simulate measurement noise. All processes are assumed to be stationary and randomness is defined by Gaussian distribution.

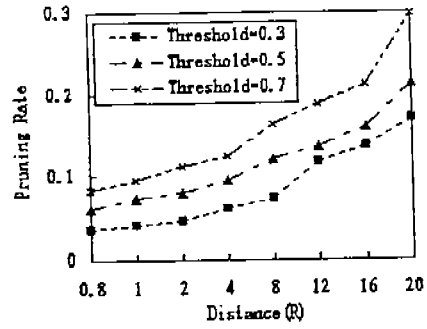
In multiple-class problems, separated processes rep-

resent each class, and at least one of the parameters has to have a different distribution for each class. On the other hand, the classes are distinguished by the mean vectors of the parameters which have random values. The problem difficulty is determined by the statistical difference between processes, which is described by the Mahalanobis distance (R) between the parameter distributions. We assumed that random variables are independent and variances are identical for each distribution.

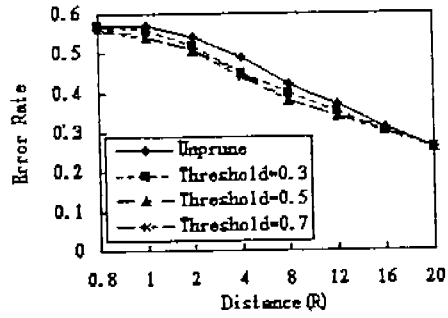
5.2. SIGNAL CLASSIFICATION AND PRUNING PERFORMANCE

Wide variety of experimental results for this random signal classification problem are described in [19, 21]. In this study, we considered two class problems with the exponential coefficient as distinguishing parameter which defined the most difficult problems [21]. Therefore, the processes used different mean values for the exponential coefficient parameter. Pattern vectors had 32 dimensions. We used networks which had a single hidden layer with 4 units. This network was found the minimum architecture [21]. For terminating the initial training, we used a root-mean-squared-error of 0.022 and a maximum number of epochs of 20,000; the maximum number of epochs during each retraining cycle was 3,000; and the total number of retraining epochs was 10,000. Learning rate was 0.01 and the momentum was 0.9. A set of 100 patterns from each class was used for training, and an independent set of 100 patterns was used for testing. For each experiment, we calculated the error rates with the test set at the ends of the initial training and the pruning algorithm.

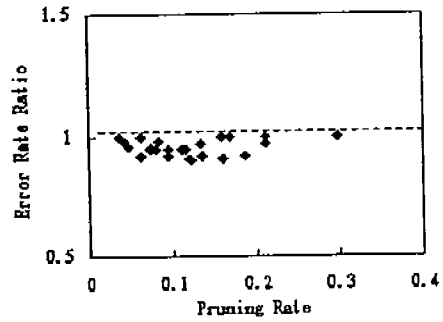
Figure 3(a) shows the error rates for the unpruned and the pruned networks versus distance with threshold of 0.3, 0.5, and 0.7 and (b) represents the corresponding pruning rates. As distance R increased, i.e., the classes become more distinct and classification is less difficult, the error rate decreased while the pruning



(a) Error Rates



(b) Pruning Rates



(c) Error Rate Ratio

(Figure 3) (a) Error rate and (b) pruning rate versus the problem difficulty (R) with various threshold values for pruning. (c) Error rate ratio of pruned networks to unpruned networks.

rate increased. This result indicates that our algorithm is consistent with the general idea that an easier problem produces a lower error and requires a simpler network. Figure 3(c), which presents Figure 3(a) and (b) in a

different way, shows the ratio of the error rate for our pruning algorithm to that for the standard algorithm as a function of pruning rate. This figure shows that our pruning algorithm demonstrated better classification performance as indicated by the ratios below 1.0. In summary, the minimal network can have some superfluous connections and elimination of those connections improves generalization. These data emphasize the importance of pruning the redundant connections.

6. CONCLUSION

We have described a simple training procedure that can eliminate superfluous connections. It was developed based on the distribution of the magnitudes of the weights and the biases, analysis of the classification boundaries, and the nonlinearity of the sigmoid function. It is a repetitive two-step training algorithm that trains the network by adjusting all weights and biases and then prunes (i.e., set to zero) the weights and biases with magnitudes less than a threshold. This algorithm is important because we can avoid the need for selecting the most efficient architecture before training begins. Instead, we can use oversized network intentionally in order to reduce the training time and the likelihood of a local minimum trap [1], and then connection pruning can be used to reduce the network complexity.

The behavior of this algorithm was justified with two Boolean functions: XOR and Rule-and-Exception. Our simple algorithm reduced the various initial architecture for XOR problem to the one which compared favorably to the theoretical minimum network. It also eliminated the connections which are not necessary to solve the Rule-and-Exception function. In pattern recognition study, we applied our algorithm to simulated random signal classification problems. Random pulse signal patterns with damped oscillatory edges were used. Our algorithm produced the results which are consistent with the idea that easier problems require simpler networks and yield lower misclassification rates. Furthermore, in comparison with the unpruned network, our algorithm produced better classification performance than the standard algorithm. We believe that our algorithm can reduce overfitting and pattern memorization problems by eliminating superfluous network parameters.

During our experimental study, we have observer that some input and hidden units can be completely deleted. This fact indicates that our algorithm can be utilized for unit level minimization and feature reduction.

Since our algorithm uses a repetitive two-step process, pruning and retraining, entire training time is generally longer than the standard training algorithm. To overcome this problem, we should concern a method for fast convergence [22, 23]. We also need further research to develop a systematic method to select the threshold value.

REFERENCES

[1] [1] D. E, Rumelhalt, J. L. McCelland, and the PDP Research Group, *Parallel Distributed Processing, Explorations in the Microstructure of Cognition, Vol. 1: Foundations.* MIT Press, 1986.

[2] A. Blumer, A. Ehrenfeucht, D. Haussler and M. Warmuth, "Learnability and the Vapnik-Chervonenkis dimension," *J. Ass. Comput. Mach.*, Vol. 36, No. 4, pp. 929-965, 1989.

[3] A. Ehrenfeucht, D. Haussler, M kearns and L. Valiant, "A general lower bound on the number of examples needed for learning," in *Proc. 1988 Workshop Computational Learning Theory*, 1988.

[4] L. Valiant, "A Theory of the learnable," *Commun. Ass. Comput. Mach.*, Vol. 27, No. 11, pp. 1134-1142, 1984.

[5] R. Reed, "Pruning Algorithms-A Survey," *IEEE Trans. Neural Networks*, Vol. 4, No. 5, pp. 740-746, 1993.

[6] S. Abe, "Learning by Parallel Forward Propa-

gation," *IJCNN International Joint Conference on Neural Networks*, Vol. III, pp. 99-104, 1990.

[7] Y. Chauvin, "A Back-Propagation Algorithm with Optimal Use of Hidden Units," in *Advances in Neural Information Processing Systems*, Vol. I. David S. Tuoretzky (Ed.), Morgan Kaufmann Publisher, pp. 519-526, 1989.

[8] M. Gutierrez, J. Wang, and R. Grondin, "Estimating Hidden Unit Number for Two-Layer Perceptron," *IJCNN International Joint Conference on Neural Networks*, Vol. II, pp. 677-681, 1989.

[9] M. Hagiwara, "Novel Back Propagation Algorithm for Reduction of Hidden Units and Acceleration of Convergence Using Artificial Selection," *IJCNN International Joint Conference on Neural Networks*, Vol. I, pp. 625-630, 1990.

[10] M. Hagiwara and M. Nakagawa, "Supervised Learning with Artificial Selection," *IJCNN International Joint Conference on Neural Networks*, Vol. II, pp. 611, 1989.

[11] Y. Hirose, K. Yamashita and S. Hijiya, "Back-Propagation Algorithm Which Varies the Number of Hidden Units," *Neural Networks*, Vol. 4, pp. 61-66, 1991.

[12] M. C. Mozer and P. Smolensky, "Skeltonization: A Technique for Trimming the Fat from a Network via Relevance Assessment," in *Advances in Neural Information Processing Systems*, Vol. I, David S. Tuoretzky (Ed.), Morgan Kaufmann Publisher, pp 107-115, 1989.

[13] J. Sietsma and R. J. F. Dow, "Neural Net Pruning-Why and How?," *IEEE International Conference on Neural Networks*, Vol. I, pp. 325-333, 1988.

[14] J. Sietsma and R. J. F Dow, "Creating Artificial Neural Networks that Generalize," *Neural Networks*, Vol. 4, pp. 67-79, 1991.

[15] M. Ishikawa, "A structural Learning Algorithm with Forgetting of Link Weights," *IJCNN International Joint Conference on Neural Networks*, *Vol.* II, pp. 626, 1989.

[16] E. Karnin, "A Simple Procedure for Pruning Back-Propagation Trained Neural Networks," *IEEE Trans. on Neural Networks*, Vol. 1, pp. 239-242, 1990.

[17] A. S. Weigend, D. E. Rumelhart and A. Huberman, "Back-propagation Weight Elimination and Time Series Prediction," in *Proc. of the 1990 Connectionist Models Summer School*, Morgan Kaufmann, pp. 65-80, 1990.

[18] A. S. Weigend, A. Huberman and D. E. Rumelhart, "Predicting Sunspots and Exchange Rates with Connectionist Networks," in *Nonlinear Modeling and Forecasting, SFI Studies in the Sciences of Complexity*, Vol. 12, Addison-Wesley, 1991.

[19] Y. Won, "Connection pruning algorithms and their comparison with the standard back-propagation algorithm," Master Thesis, University of Missouri, 1991.

[20] R. P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, pp. 4-22, April 1987.

[21] D. S. Park, "Relationship Between the Performance of Multilayer Neural Network Pattern Classifiers and the Statistics of Pattern Generating Processes," Ph. D. Dissertation, University of Missouri - Columbia, 1990.

[22] A. Jacobs, "Increased Rates of Convergence through Learning Rate Adaptation," *Neural Networks*, Vol. I, pp. 295-307, 1988.

[23] J. Sun, W. I. Grosky, and M Hassoun, "A Fast Algorithm for Finding Global Minima of Error Functions in Layered Neural Networks," *IJCNN International Joint Conference on Neural Networks*, Vol. I, pp 715-720, 1990.

원 용 관

1987년 한양대학교, 전자공학과 졸업(학사)
1991년 미주리 주립대학교, Electrical and Computer Engineering(석사)
1995년 미주리 주립대학교, Electrical and Computer Engineering(박사)
1987년~88년 금성 통신 근무
1992년~93년 U. S. Postal Office 지원 *수기* 문자 인식
1994년~95년 U. S. Air Force 지원 자동 목표물인식 연구
1995년 11월~현재 한국전자 통신 연구소, 인공지능 연구실
관심분야: Image Processing, Mathematical Morphology, Neural network, Fuzzy logic, 자동 목표물 인식, Virtual Reality, *수기* 문자 인식

민 병 의

1982년 한양대학교 졸업(학사)
1984년 한국 과학 기술원 전기 및 전자공학과 졸업(석사)
1992년 한국 과학 기술원 전기 및 전자공학과 졸업(박사)
1984년~87년 대림산업 기술 연구소
1987년~현재 한국 전자 통신 연구소, 인공지능 연구실, 실장
관심 분야: 멀티미디어 시스템, 에이전트, Virtual Reality