은닉지식 추출을 이용한 신경회로망 정제
(Neural Network Refinement using Hidden Knowledge Extraction)

김 현 철 †
(Hyeoncheol Kim)

요 약 신경회로망 구조의 정제(精製)는 회로망의 일반화능력이나 효율성의 관점에서 중요한 문제이다. 본 논문에서는 feed-forward neural networks로부터 은닉지식을 추출하는 방법을 사용하여 네트워크 재구성을 통한 정제방법을 제안한다.

먼저, 효율적인 if-then rule 추출방법을 제시하고 그 추출된 룰들을 사용하여 룰기반 네트워크로 변환하는 과정을 보여준다. 생성된 룰기반 네트워크는 fully connected network에 비하여 상당히 축소된 연결 복잡도를 가지게 되며 일반적으로 더 우수한 일반화능력을 가지게 된다. 본 연구는 도메인 지식이 없이 데이타만 사용하여 어떻게 정제된 룰기반 신경회로망을 생성하고 있는가를 보여준다. 도메인 데이타들에 대한 실험결과도 제시하였다.

Abstract How to refine a neural network structure has been an important issue in designing a feedforward network. In this paper, we propose a new approach to refine a feedforward neural network using knowledge extraction from the network.

An efficient method for extracting hidden rules of a feedforward network is presented. The extracted rules are mapped into a rule-based neural network which represents the knowledge of the original neural network. The rule-based network explains its behaviour by internal rules and often provides better generalization performance. Through this process, a cumbersome fully connected neural network can be transformed into an efficient rule-based connectionist network of much reduced size. The main contribution is that we show how the refined rule-based network is generated from data by neural adaptation without resource to any prior domain knowledge. Empirical results are shown.

1. Introduction

Refinement of neural network connection structure is a strategy for improving its generalization and efficiency. Fully-connected networks with arbitrary number of hidden nodes often suffer over-fitting problem caused by overtraining and unbalance training data set which degrades the generalization capability of standard multi-layered networks. There has been studies to find the right complexity of the network such as skeletonization pruning[1], weight decay[2], gain decay[3], etc. Those methods try to reduce the hidden layer size down to an optimal size.

Given any prior domain knowledge (or rules), we can use the domain knowledge to determine the initial structure of the neural network. The knowledge-based neural networks have been shown better network structures and thus better generalization [4,5]. However, the knowledge-based neural network can not be constructed without prior domain knowledge (or rules) and thus, this network has been used for domain knowledge refinement system rather than network refinement system [6, 7].

In this paper, we propose a new approach to the network refinement which involves rule extraction from a fully-connected feed-forward neural network and reconstruction of a rule-based neural network using the extracted rules only (i.e., without any

† 정 회 원 : 고려대학교 컴퓨터교육과 교수
hkim@comedu.korea.ac.kr
논문접수 : 2000년 2월 24일
심사완료 : 2000년 9월 22일

prior domain knowledge). An efficient algorithm to extract hidden knowledge from a neural network is presented. We show how this method is applied to network refinement and the resulting network with much reduced size often provides better generalization performance.

2. Rule-Based Neural Networks

In rule-based connectionist networks (RBCN) [8, 9], its connection topology is determined by symbolic rules as shown in Figure 1 and thus, the complexity of the network is determined by the complexity of the rules used. The network constructed by the rules simulates a rule-based system if it is not trained. After the network topology is determined, weights are initialized properly according to given knowledge. Depending on the type of activation function chosen, different weight initialization schemes are adopted. See, for example, the KBCNN model [8] and the KBANN model [7,10]. In this paper, we adopt a multi-layered feed-forward backpropagation network architecture whose topology is determined by *if-then* rules and activation function is sigmoid.

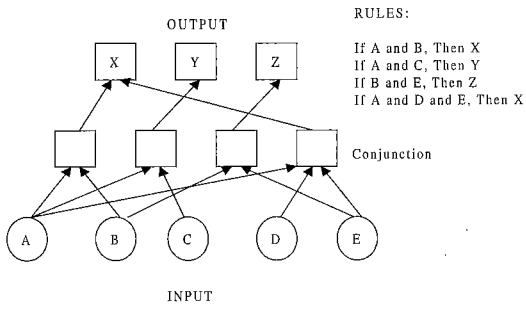

Fig. 1 An example of a rule-base dconnectionist network [2].

3. Rule Extraction for Single Nodes

A rule generated from a neural network has the form of *"if the premise, then the conclusion."* The premise is composed of a number of positive and negative attributes and so is the conclusion. In the basic form of a rule, the rule's premise is limited to a conjunction of attributes and the rule's conclusion is limited to a single attribute. However, the presence of multiple rules with same conclusion represents disjunction. A rule with a conjunction of conclusions is represented by multiple rules with same premise but different conclusions. Quality of a rule is evaluated with a few criteria. First of all, a rule should be *valid*. The validity condition for a rule is defined as follows. Whenever the rule's premise holds, so does its conclusion in the presence of any combination of the values of attributes not referenced by the rule[9]. Other criteria include *accuracy* (specificity or goodness of fit) and *generality* (simplicity). Accuracy is about how often the rule is classified correctly. Generality is about how often the premise of a rule occurs. It is also related to coverage of the rule. As the rule gets simpler (i.e., shorter), it covers more instances in the input domain.

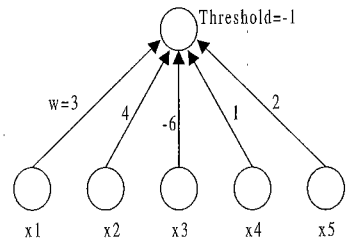

Fig. 2 An individual node with five incoming connection weights and a threshold.

Figure 2 gives an example to illustrate *valid* rule generation. At each non-input unit of a trained network, n incoming connection weights and a threshold are given. Rule extraction at the unit finds a set of incoming attribute combinations that are valid and maximally-general (i.e., size of each combination is as small as possible). For example, a combination (x_1, x_2) is a valid rule because its total-input (i.e., *3+4=7*) is always greater than the threshold (i.e., *-1*) regardless of the values of other incoming units such as x_3, x_4 and x_5. This combination is converted into a valid rule "If x1

and x2, then y". Another example is (not x_3) which is also valid.

Even though this procedure is very simple, search space is increases exponentially with the number of incoming units. Given n binary attributes, there are $2^n(=\sum_{l=0}^{n} C_l^n)$ possible combinations. For each combination, its validity testing adds another complexity 2^l and thus, $3^n(=\sum_{l=0}^{n} 2^l C_l^n)$ possible rules in the rule space.

There has been many other studies in extraction of valid and general rules efficiently [4,11,12,13,14, 15]. The complexity for validity testing can be reduced to $O(1)$ by a simple method that tests only one encode with the lowest total-input [8,9,11]. Therefore rule search space is reduced to 2^n from 3^n. Tree-based search uses a tree structure where each node is one of the 2^n combinations. At depth l in the search tree, length (i.e., the number of attributes) of combinations is l. The tree-based search begins with a root node (i.e., length 0 or the shortest length combination) to a leaf node (i.e., length n or the longest length combination), checking its validity. If it finds a valid rule combination, it stops generating children nodes and takes it as a rule with the following property: *"if a combination of a node is a valid rule, combinations of its descendant nodes are eliminated from the search tree space."*, which reduces search space. For finding the best rule (i.e., a valid combination with the shortest length), the worst case $O(2^n)$ occurs when the rule is a leaf node combination of length l. Fu's KT[9,11] algorithm is an improved tree search algorithm which reduces search space and memory space heuristically. The KT algorithm reduces the search space by dividing the n incoming attributes into p positive and q negative attributes (st. $p+q=n$). The basic idea of the separation derives from the fact that the complexity of an exponential space is greater than the sum of complexities of its sub-spaces. That is, $x^l \geq (a \cdot x)^l + (b \cdot x)^l$ where $l>1$, $a<1$, $b<1$ and $a+b=1$. Even though this approach reduces the complexity in most cases, the worst case complexity remains same $O(2^n)$ if $p=n$ (i.e., no negative attributes).

We introduce a different search space structure and computationally efficient heuristics in this paper. The algorithm involves the following three procedures:

① Attribute Contribution Scoring: For each attribute, its contribution to a candidate rule is calculated.

② Attribute Sorting: Attributes are sorted in descending order according to their contribution scores.

③ Rule Searching: With the attributes sorted by their contribution scores, valid and maximally general rules are to be searched.

The contribution score of each attribute is defined by the amount of total-input increase (or decrease) when the attribute is added to a candidate rule. For binary attribute cases, the contribution score of an attribute x_i is defined as $|w_i|$ where the w_i is the connection weight of x_i.

After the contribution scores are calculated, attributes are sorted in descending order by their scores as follows: $a_1, a_2, a_3, \ldots a_n$. Then we list the combinations of length l in the following order.

$$(a_1, a_2, \ldots a_l), \ldots (a_1, a_2, \ldots a_n), \ldots (a_1, a_3, \ldots a_{l+1}), \ldots (a_{n-l+1}, \ldots a_n) \quad (1)$$

At each depth l in the search tree, the number of combinations is C_l^n and they are listed from the left to the right as defined in (1). For example, we have 10 incoming attributes and they are sorted by their contribution scores as follows: $a_1, a_2, a_3, \ldots a_{10}$. Figure3 illustrates a search tree based on the ordered attributes. At depth 3, the *120* (i.e., C_3^{10}) combinations are listed in the following order: $(a_1, a_2, a_{k=3..10}), (a_1, a_{j=3..9}, a_{k=(j-1)..10}), (a_{i=2..8}, a_{j=(i+1)..9}, a_{k=(j+1)..10})$.

Search begins level by level from root to leaf node. This structure provides three useful heuristics:

① At each level l, the left-most node (the first combination) holds the highest total-input and the last one holds the lowest. Therefore, it is

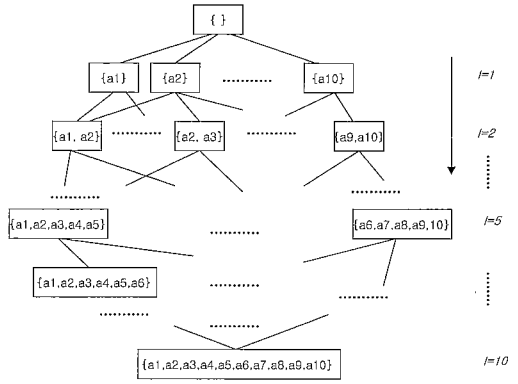

Fig. 3 Rule search tree structure with ordered attributes a_1, a_2, ... a_{10}.

straightforward to find the best rule (i.e., valid combination with the shortest length). It checks only the first node at each depth until validity condition is satisfied. If the first combination is not valid, the other combinations at the depth are not valid either and thus eliminated from the search tree instantly. Therefore, finding the best rule costs only $O(n)$ worst case.

② For finding more than one rule, for example, b $(b>1)$ best rules, search begins from the depth, d, of the best rule found at ①, eliminating the search space in depth 0 through $d-1$. At depth d, it tests validity of each node from left to right. The combination order at each depth provides another useful heuristic to reduce search complexity. For example, let the d be 3 and the number of attributes, n, be 10. Then the number of possible combinations is $C_3^{10}=120$. Consider a combination $R=(a_1, a_3, a_6)$. Then, total-input of R is greater than the ones of $(a_1, a_3, a_7), \ldots (a_1, a_3, a_{10})$. If the R is not valid, the $\sum_{k=6+1}^{n} 1=4$ combinations are not valid either. In the same way, total-input of R is greater than the ones of $(a_1, a_{j>3}, a_{k\geq \max(6, j+1)})$ and $(a_{i>1}, a_{j\geq \max(3, i+1)}, a_{k\geq \max(6, j+1)})$, eliminating 20 and 80 combinations, respectively. Therefore, if the R is not valid, this heuristic eliminates 104 other combinations from the search space instantly.

③ Another heuristic is from the tree-based search property described before. At depth l, if a combination is valid, its child combinations at depth $l+1$ are eliminated from the search space because they are subsumed to the parent.

The three heuristics presented above reduce search space significantly. Contribution scoring procedure costs $O(n)$ and sorting costs $O(n\log n)$. Rule searching procedure costs $O(n)$ for finding one best rule and $O(2^{n-1})$ worst case for b rules.

4. Empirical Results

Three data sets from public domains were used in experiments: promoter, hepatitis and iris. The experimental procedure is shown in Figure 4. We compared the performance of the trained standard network and the trained RBCN; the latter was derived from the former by rule extraction, as described. Generalization capability was evaluated by cross-validation: Each data set is divided into two independent subsets, for training and testing, respectively.

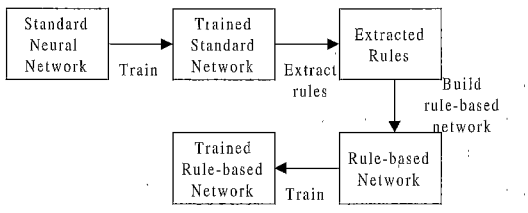

Fig. 4 The experimental procedure

The promoter domain has 106 instances. Each instance consists of a DNA nucleotide string of four base types: A(adenine), G(guanine), C(cytosine), and T(thymine). Each instance string involves 57 sequential nucleotides. An instance is a positive instance if the promoter region is present in the string; else it is a negative instance. There are 53 positive and 53 negative instances. In the Hepatitis domain, there are 155 instances, each described by 19 features: six continuous features and thirteen nominal features. The Fisher's iris data set contains 3 classes with 50 instances each, and each class refers to a type of flower, the iris.

One class is linearly separable from the other two which are not linearly separable from each other. Each instance in the data set is described by four continuous features. For the experiments here, each continuous feature is discretized to three interval attributes, resulting in a total twelve binary input attributes.

Table 1 shows each domain's network configurations which specify the number of units in each layer in the form of (input layer)-(hidden layer)-(out layer) for standard networks and (input layer)-(first hidden layer)-(second hidden

layer)-(third hidden layer)-(output layer) for RBCNs. Table 2 shows a set of production rules that are extracted from an original network trained with iris data set2. For this set, we extracted two rules from every non-input node: one confirming rule and one disconfirming rule with highest total-sum. Figure 5 illustrates the RBCN constructed with the 17 extracted rules. The initial weights of each input connections from positive attributes is set to around $1/p$ (or $-1/q$ for the connection from negated attributes) where the p (or q) is the number of positive (negative) attributes in the rule's premise. The initial threshold of the corresponding conjunction unit is set to about *0.2*. The weights from conjunction nodes to output or hidden nodes are set to strong values (e.g. 0.5). The performance comparison is shown in Table 3. The result of the promoter domain is quite promising because the RBCN improvesgeneralization with 20 times smaller number of connections. In the hepatitis domain, the RBCN has better generalization with only 37.7% of the size of

Table 1 Network configuration

Domain data		Network configuration	
		RBCN	Original NN
Promoter	set1	228-6-5-5-1	228-14-1
	set2	228-13-7-18-1	
Hepatitis	set1	34-39-18-9-2	34-20-2
	set2	34-30-17-11-2	
Iris	set1	12-12-6-6-3	12-6-3
	set2	12-11-6-6-3	

Table 2 The extracted rules from the trained standard neural network on the iris data set2. 11 rules from hidden nodes and 6 rules from output nodes.

R0	If x8 and x11	Then not h0
R1	If x5 and x7 and x10	Then h0
R2	If x1 and x3 and x7 and x11	Then not h1
R3	If x0 and x5 and x6 and x9	Then h1
R4	If x2 and x6 and x9	Then not h2
R5	If x2 and x5 and x7 and x10	Then not h3
R6	If x8 and x11	Then h3
R7	If x8 and x11	Then not h4
R8	If x5 and x7 and x10	Then h4
R9	If x5 and x7 and x10	Then not h5
R10	If x8 and x11	Then h5
R11	If not h1 and h2	Then not y0
R12	If h0 and h1 and not h2 and not h3 and not h4 and h5	Then y0
R13	If not h0 and not h2 and not h4 and h5	Then not y1
R14	If h2 and h4 and not h5	Then y1
R15	If h0 and h4	Then not y2
R16	If not h0 and not h1 and not h2 and h3 and not h4 and h5	Then y2

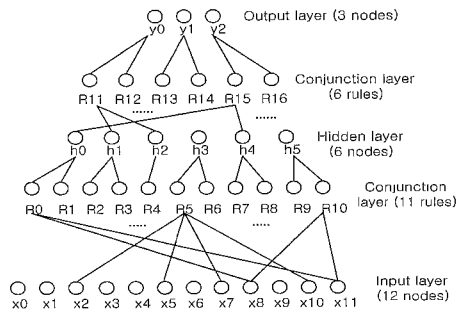

Fig. 5 Rule-based connectionist network constructed from the extracted rules in Table 2. The number of connections in this network is 98.

original network. The iris domain has a small number of input attributes, so the network size is small. The same generalization was obtained with much smaller number of connections.

5. Conclusion

In contrast to related work on knowledge-based

Table 3 Generalization performance and the number of connections of RBCNs and original neural networks on the three domains.

Domains		RBCN			Original NN		
		set1	set2	avg	set1	set2	avg
Promoter	Generalization (%)	84.91	86.79	85.85	81.11	84.91	83.02
	Connections	96	217	156.5	3221		
Hepatitis	Generalization (%)	96.10	96.15	96.13	94.81	93.59	94.19
	Connections	272	288	280	742		
Iris	Generalization (%)	98.67	98.67	98.67	98.67	98.67	98.67
	Connections	108	98	103	234		

neural networks, this work shows how to generate a rule-based connectionist network on the data without resource to any prior domain rules. This process involves extracting rules for each hidden and output units in the neural network and then mapping these extracted rules into a rule-based connectionist network. The main issues addressed are the complexity of rule generalization. The rule extraction heuristics presented in this paper is computationally efficient compared to previous ones. The results of this study suggest that a fully-connected neural network can be converted into an efficient rule-based connectionist network of reduced size. Yet, its feasibility has much to do with whether the domain is governed by rules.

References

[1] Mozer, M. and Smolensky, P., "Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment, *Advances in Neural Information Processing System 1*, (editor: Touretzky, D.S.) 1989

[2] Krogh, A. and Hertz, J.A., A Simple Weight Decay Can Improve Generalization, *Advances in Neural Information Processing Systems*, Vol. 4, pp950-957, 1992

[3] Kruschke, J.K., and Movellan, J.R., Benefits of Gain: Speeded Learning and Minimal Hidden Layers in Backpropagation Networks, *IEEE Transactions on Systems, Man and Cybernetics*, 21(1), January, 1991

[4] Fu, LiMin and Kim, Hyeoncheol, Abstraction and Representation of Hidden Knowledge in an Adapted Neural Network, *Technical Report*, CISE, University of Florida, October 1994.

[5] Kim, Hyeoncheol and Fu, LiMin, Generalization and Fault Tolerance in Rule-based Neural Network, *In Proceedings of IEEE International Conference on Neural Networks*, volume 5. pp. 1550-1555, 1994

[6] Opitz, D.W., and Shavlik, J.W., Using Genetic Search to Refine Knowledge-Based Neural Networks, *Machine Learning: Proceedings of the 11th International Conference*, 1994

[7] Towell, G.G. Shavlik, J.W. and Noordewier, M.O., Refinement of Approximate Domain Theories by Knowledge-based Neural Networks, *Proceedings AAAI-90*, pp861-866, 1990

[8] Fu, LiMin, Knowledge-based connectionism for revising domain theories. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(1), pp. 173-182. 1993

[9] Fu, LiMin, *Neural Networks in Computer Intelligence*. chapter 14, McGraw Hill, Inc., New York. 1994

[10] Towell, G.G and Shavlik, J.W., Knowledge-Based Artificial Neural Networks, *Artificial Intelligence*, vol. 70, pp119-165. 1994

[11] Fu, LiMin, Rule generation from neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(8), pp. 1114-1124. 1994

[12] Gallant, S.I. Connectionist Expert Systems. *Communications of the ACM*, 31(2), pp. 152-169. 1988

[13] Setino, Rudy and Liu, Huan, Understanding neural networks via rule extraction. *In Proceedings of the 14th International Joint Conference on Artificial Intelligence*, vol.1, pp. 480-485, 1995

[14] Setino, Rudy and Liu, Huan, Symbolic Representation of Neural Networks, *Computer*, pp. 71-77, 1996

[15] Taha, I.A. and Ghosh, J., Symbolic Interpretation of Artificial Neural Networks, *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 3. 1999.

김 현 철

1988년 고려대학교 전산과학과 학사. 1990년 Univ. of Missouri-Rolla, Computer Science (M.S.). 1998년 Univ. of Florida, Computer & Information Sciences (Ph.D.). 1998년 미국 GTE Data Services, Inc. R&D Staff. 1998년 ~ 1999년 삼성SDS 책임연구원. 1999년 ~ 현재 고려대학교 사범대학 컴퓨터교육과 조교수. 관심분야는 Machine Learning, Data Mining, Internet-Based Instruction.