학습된 지식의 분석을 통한 신경망 재구성 방법
(Restructuring a Feed-forward Neural Network Using Hidden Knowledge Analysis)

김 현 철 †
(Hyeoncheol Kim)

요 약 다층신경회로망 구조의 재구성은 회로망의 일반화 능력이나 효율성의 관점에서 중요한 문제로 연구되어왔다. 본 논문에서는 신경회로망에 학습된 은닉 지식들을 추출하여 조합함으로써 신경회로망의 구조를 재구성하는 새로운 방법을 제안한다. 먼저, 각 노드별로 학습된 대표적인 지역 규칙을 추출하여 각 노드의 불필요한 연결구조들을 제거한 후, 이들의 논리적인 조합을 통하여 중복 또는 상충되는 노드와 연결구조를 제거한다. 이렇게 학습된 지식을 분석하여 노드와 연결구조를 재구성한 신경회로망은 처음의 신경회로망에 비하여 월등히 감소된 구조 복잡도를 가지며 일반적으로 더 우수한 일반화 능력을 가지게 됨을 실험결과로서 제시하였다.

키워드 : 지식기반 신경망, 규칙추출, 신경망 정제

Abstract It is known that restructuring of a feed-forward neural network affects generalization capability and efficiency of the network. In this paper, we introduce a new approach to restructure a neural network using abstraction of the hidden knowledge that the network has learned. This method involves extracting local rules from non-input nodes and aggregation of the rules into global rule base. The extracted local rules are used for pruning unnecessary connections of local nodes and the aggregation eliminates any possible redundancies and inconsistencies among local rule-based structures. Final network is generated by the global rule-based structure. Complexity of the final network is much reduced, compared to a fully-connected neural network and generalization capability is improved. Empirical results are also shown.

Key word : Know ledge-Based Neural Network, Rule Extraction, Neural Network Pruning

1. Introduction

Fully-connected neural networks with arbitrary number of hidden nodes often suffer over-fitting problem which degrades its generalization capability. Refinement of connection structure of a neural network is one of the strategies for improving its generalization and efficiency. There has been studies to find the proper 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. In this paper, we introduce a method to get proper complexity using rule-based structure.

Given any prior domain knowledge (or rules), the domain knowledge can be used to determine the initial structure of a neural network. The knowledge-based neural networks provide better network structures and thus better generalization[4, 5]. However, the knowledge-based neural network can not be constructed without prior domain knowledge and thus, this network has been used for domain knowledge refinement rather than network refinement[6, 7]. Kim[8] introduced knowledge-based refinement of neural network structure without resource to any prior domain knowledge. His method involves node-based rule

† 정 회 원 : 고려대학교 사범대학 컴퓨터교육과 교수
hkim@comedu.korea.ac.kr
논문접수 : 2001년 8월 24일
심사완료 : 2002년 1월 28일

extraction from non-input nodes and restructuring connections of each node using the local rules. The node-based method reduces number of connections of each node, but not nodes and layers of the network. The resulting network is not slim enough and contains possible redundancies and inconsistencies in the network configuration.

In this paper, we propose a new approach that reconstructs nodes and layers as well as connections of the network. This method involves extraction of rules from each non-input node and aggregation of the local rules to generate a composite rule base for the network. The final rules are used for constructing knowledge-based structure for the network. This network-based restructuring reduces network complexity significantly without losing performance, compared to the node-based method.

2. Node-Based Restructuring Using Extracted Rules

2.1. Rule Extraction from Single Nodes.

A rule 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. As the rule gets simpler(i.e., shorter), it covers more instances in the input domain.

There has been many other studies in extraction of valid and general rules from neural networks[10, 11, 9, 12, 4, 13, 14, 15, 16, 17]. Rule extraction from a non-input node searches maximally-general (i.e. the shortest size of premise) and valid combinations of incoming connections. The rule extraction is computationally expensive since the rule search space is increased exponentially with the number of input attributes. If a node has n incoming nodes, there are 3^n possible combinations. Many studies tried to reduce the search space efficiently[11, 9, 12, 4, 8, 14, 15]. Kim[14] introduced a computationally efficient algorithm called OAS (Ordered-Attribute Search). The OAS algorithm assigns so-called contribution scores to each attributes and sorts them in descending order by the scores. Search tree is organized by the ordered attributes and three useful heuristics are introduced. The heuristics reduce the search space significantly. In this paper, the OAS is used for extraction of one or two best rules from each node and its time complexity is $O(n \log n)$.

2.2. Rule-Based Structure for Local Nodes

In rule-based neural networks (RBNN)[11, 9, 7, 18, 8], its connection topology is determined by symbolic rules and thus the complexity of the network is determined by the complexity of the rules used. After rules are extracted from individual non-input nodes, the rules are mapped into a neural network architecture as shown in Figure 1(a) and 1(b). Note that a hidden unit is introduced to explicitly represent the conjunction of the conditions in a rule's premise. Such a hidden unit is called a conjunction unit. In other words, to map a rule, we need a three-level construct: The first level consists of the attributes of the *if* part, the next level consists of the conjunction of these attributes to form the premise, and the third level consists of the consequent (in the *then* part). If we omit the conjunction level and link the attributes directly to the consequent, it will cause a problem when there were multiple rules because some combinations of the attributes involved in different rules may also

activate the target concept. To avoid these possible unintended combinations, each rule's premise is assigned a conjunction unit which activates the target concept disjunctively[19].

As shown in Figure 1(b), the extracted rules are organized in a two-level hierarchy, which are mapped into a neural network with three hidden layers, that is, one disjunction and two conjunction layers. The number of conjunction units is determined by the number of rules and the number of hidden layers is determined by the number of levels in the rule hierarchy. The RBNN provides better understanding and often better generalization performance than a standard fully-connected network [5, 8]. Kim and Fu [4, 8] showed that the RBNN can be constructed by node-based rule extraction.

3. Global Knowledge-Based Restructuring using Logical Aggregation

In this paper, we introduce RBNN2 (Rule-Based Neural Network 2) which is derived from the RBNN. The rules in multi-level hierarchy in the RBNN are logically aggregated to form a composite rule set for the neural network as shown in Figure 1(c). It rewrites rules to eliminate the symbols which refer to hidden units but are not predefined in the domain. Thus multiple hidden layers are replaced by one conjunction layer. In the process, any possible redundancies, subsumption or inconsistencies are removed.

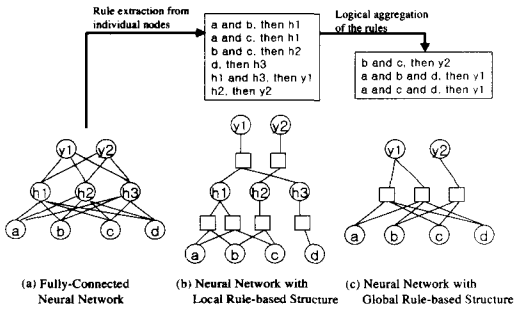

Figure 1 Restructuring Process

The purpose of the aggregation is to convert node-based rule structure of RBNN into network-based rule structure which is more compact and still logically same. The algorithm is given below:

1. Find a rule (call R_y) with output layer.
2. If antecedent of the R_y contains any undefined symbol, find all rules (call R_h's) whose consequents reference the symbol.
3. Replace the symbol in R_y by the antecedent of each R_h to generate a new rule.
4. Logical adjustment for the new rewritten premise of the R_y.
5. Go to step 1.

Redundancy can be defined by appearance (syntactic redundancy) or by meaning (semantic redundancy). In the first sense, if two rules are identical, then they are redundant with respect to each other. In the second sense, if two rules are mutually replaceable (despite their appearance), they are mutually redundant. Given a rule-based system, a rule is redundant if it can be deleted without affecting the system performance. Intermediate concepts associated with hidden units often generate redundancies. During aggregation process, multi-level rules are rearranged into one-level rules and any possible redundancies or inconsistencies are removed. For example,

If a and b and not c, then h1
If a and b and not c, then h2
If h1, then y1
If h2, then y1

The four rules in two-level hierarchy can be aggregated as follows:

If h1 or h2, then y1
⇒ If (a and b and not c) or (a and b and not c), then y1
Thus, "If a and b and not c, then y1"

The set of four rules are logically same as the final set of one rule.

Another example with an input feature which are multi-valued. Suppose that a feature *Color* have three values such as (r)ed, (b)lack and (g)reen and each value is encoded as an input attribute. They

are mutually exclusive in the premise of a rule.

If a and not r, then h1

If a and not b, then h2

If h1, then y1

If h2, then y1

Logically they can be rewritten as follows.

If h1 or h2, then y1

⇒ If (a and not r) or (a and not b), then y1

⇒ If a and (not r or not b), then y1

⇒ If a and (b or g or r), then y1

Thus, "If a, then y1"

The rewritten rules also satisfy the validity condition. A rule R_{y1}: *"If h1 and h2, then y1"* is a conjunction of the rules of R_{h1}: *"If premise, then h1"* and R_{h2}: *"If premise, then h2"*. Since certainties of the two rules are close to *1*, certainty of their conjunction is also close to *1*. However, if product of certainties of R_{h1} and R_{h2} is not greater than *0.5*, we discard the rewritten rule.

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 in the following way. Suppose that a rule's premise involves p positive attributes and q negated attributes. The initial threshold of the corresponding conjunction unit is set to about *0.2*, and the initial weight of each input connection from positive attributes is set to around $1/p$ (or $-1/q$ for the connection from negated attributes). The weights from conjunction nodes to output or disjunction nodes are set to belief values attached to the rules of the conjunction nodes or set to strong values (e.g. *0.5*) [9, 11]. The initial RBNN and RBNN2 are trained by back propagation procedure with training data set.

4. Experimental Results

Experiment procedure is as follows. A fully-connected neural network is trained. Then local rules are extracted from individual non-input nodes of the network. The rules are used to construct a rule-based neural network (i.e. RBNN) in which connections of each node are restructured according to the rules. Then the rules are aggregated to eliminate intermediate nodes and connections of the rule-based network and the rewritten rules are used to construct the final network(i.e., RBNN2) that represents network-based abstraction of what the original network has learned. We compare network complexity and generalization performance of the original network, RBNN and RBNN2. Network complexity is evaluated with the number of connections and layers of each network. Generalization capability is evaluated by *two-fold cross-validation*: a domain data set is divided into two independent subsets, and each in turn is used for testing while the other for training. The procedure is repeated twice so that in the end, every instance has been used exactly once for testing. For each testing, percentage of correctly classified instances is calculated.

Two data sets from public domains are used in experiments: iris and promoter data sets. The Fisher's iris data set contains three classes with 50 instances each. 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 12 binary input attributes. Table 1 shows performance comparison of the three different types of network. The RBNN2's generalization remains same (or better) with much reduced network complexity.

Table 1 Iris: Performance comparison of three different types of network. The number of hidden nodes of original network is 6.

	# of layers	Average # of connections	Average generalization(%)
Original Network	3	243	97.7
RBNN	5	103	98.7
RBNN2	3	24	98.7

The promoter domain has 106 instances: 53 instances of them belong to promoter class and the

rest 53 belong to non-promoter class. Each instance consists of a DNA nucleotide string of four base types: A(adenine), G(guanine), C(cytosine), and T(thymine). Each instance string is comprised of 57 sequential nucleotides. Table 2 shows network complexity and performance for the promoter domain. The RBNN and RBNN2 improves generalization with only *1/14* and *1/77* times the number of connections of the standard fully-connected neural network, respectively.

The RBNN2 network size is much reduced compared to the original network or the RBNN without losing its generalization capability.

Table 2 Promoter: Performance comparison of three different types of network. The number of hidden nodes of original network is 4.

	# of layers	Average # of connections	Average generalization(%)
Original Network	3	921	81.1
RBNN	5	64	82.1
RBNN2	3	12	84.9

5. Conclusion

Refining the structure of a neural network provides better understanding and often better generalization. The RBNN2 introduced in this paper is a new method to prune and refine the network structure by abstracting knowledge that the network has learned.

It involves extracting node-based local rules from non-input nodes of a trained neural network and then aggregating them into a set of global rules for the network. Then, the network is restructured by the global rules. This method is network-based abstraction that reconstructs nodes and layers as well as connections of the network.

This paper is related to our previous work on rule extraction from individual nodes [4, 14] and rule-based neural networks [4, 5, 8] but with emphasis on more efficient network-based knowledge structure. The main contributions are as follows. We show that the rule-based neural network (RBNN) can be refined better by logical aggregation of local rules extracted from individual nodes. The aggregation eliminates any possible redundancies and inconsistencies between local rule-based structures and constructs global knowledge-based structure. The new network (i.e., RBNN2) provides better understanding and much smaller complexity without losing performance. The network is generated without resource to any prior domain knowledge.

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 (1992) pp.950-957, 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*, Vol.21(1,) 1991.

[4] Fu, LiMin and Kim, Hyeoncheol., "Abstraction and Representation of Hidden Knowledge in an Adapted Neural Network," *unpublished*, CISE, University of Florida, 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*, Vol.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, Geoffrey G., Shavlik, Jude W. and Noordewier, M.O., "Refinement of Approximate Domain Theories by Knowledge-based Neural Networks," *Proceedings AAAI-90*, pp.861-866, 1990.

[8] Kim, Hyeoncheol., "Neural Network Refinement

using Hidden Knowledge Extraction," *Journal of KISS: Software and Applications*, Vol.27 pp.1082-1087, 2000.

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

[10] Andrews, Robert, Diederich, Joachim, Tickle, Alam B., "Survey and critique of techniques for extracting rules from trained artificial neural networks," *Knowledge-Based Systems*, Vol. 8(6) pp.373-389, 1995.

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

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

[13] Gallant, S.I., "Connectionist expert systems," *Communications of the ACM*, Vol. 31(2) pp.152-169, 1988.

[14] Kim, Hyeoncheol., "Computationally Efficient Heuristics for If-Then Rule Extraction from Feed-Forward Neural Networks," *Lecture Notes in Artificial Intelligence*, Vol. 1967 pp.170-182, 2000.

[15] Setino, Rudy, Liu, Huan, "Understanding neural networks via rule extraction," *Proceedings of the 14th International Conference on Neural Networks*, pp.480-485, 1995.

[16] Taha, Ismali A. and Ghosh, Joydeep, "Symbolic interpretation of artificial neural networks," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11(3) pp.443-463, 1999.

[17] Towell, Geoffrey G. and Shavlik, Jude W., "Extracting refined rules from knowledge-based neural networks," *Machine Learning*, Vol. 13(1), 1993.

[18] Towell, Geoffrey G. and Shavlik, Jude W., "Knowledge-Based Artificial Neural Networks," *Artificial Intelligence*, Vol 70, pp.119-165, 1994.

[19] Fu, LiMin., "Introduction to knowledge-based neural networks," *Knowledge-Based Systems*, Vol. 8(6) pp.299-300, 1995.

김 현 철

1988년 고려대학교 전산과학과 학사. 1990년 University of Missouri-Rolla 전산학 석사. 1998년 University of Florida 전산정보학 박사. 1998년~1999년 미국 GTE Data Services, Inc.와 삼성SDS 근무. 1999년 ~ 현재 고려대학교 사범대학 컴퓨터교육과 조교수. 관심분야는 기계학습이론, 데이타마이닝, 인터넷기반교육시스템