경영정보학연구
제7권 1호
1997년 6월

영향도에 기초한 의사결정유형분석 구현을 위한 신경망 응용

Kyung Sam Park*, Jae Kyeong Kim** and Hyung Je Yun***

Applied Neural Net to Implementation of Influence Diagram Model Based Decision Class Analysis

This paper presents an application of an artificial neural net to the implementation of decision class analysis (DCA), together with the generation of a decision model influence diagram. The diagram is well-known as a good tool for knowledge representation of complex decision problems. Generating influence diagram model is known to in practice require much time and effort, and the resulting model can be generally applicable to only a specific decision problem. In order to reduce the burden of modeling decision problems, the concept of DCA is introduced. DCA treats a set of decision problems having some degree of similarityz as a single unit. We propose a methoa utilizing a feedforward neural net with supervised learning rule to develop DCA based on influence diagram, which method consists of two phases: Phase I is to search for relevant chance and value nodes of an individual influence diagram from given decision and specific situations, and Phase II elicits arcs among the nodes in the diagram. We also examine the results of neural net simulation with an example of a class of decision problems.

* Visiting Scholar, MSIS Department, The University of Texas, Austin TX, U.S.A.
** Assistant Professor, Department of MIS, Kyonggi University, Suwon, 442-760
*** Senior Consultant, Consulting SSU, LG EDS, Seoul 150-010

이 논문은 1997년 3월 19일 접수하여 1차 수정을 거쳐 1997년 6월 11일 게재확정되었습니다.

I. INTRODUCTION

In contrast to evaluating a decision problem, there has been little research to formulate the decision problem. The formulation implies in this paper the construction of a decision model, such as an influence diagram (ID) originally designed for representing and solving complex decision problems [Howard & Matheson, 1984]. The diagram is a useful tool for decision analysis because of its perspicuity in graphically displaying both the variables of a decision problem and their relationships. The model construction is in practice known to be a most complicated and burdensome process. Furthermore, the resulting model is generally applicable to one specific decision problem [Reed, 1989; Kim, 1991].

From this point of view, a set of decisions having some degree of similarity among them can be treated as a single unit in order to reduce the burden of decision structuring. Holtzman [1989] defines this unit as a *class of decisions* and the corresponding collective analysis as a *decision class analysis* (DCA). He describes that whereas the end of result of an individual analysis is a decision, the end of DCA is an individual decision analysis, a decision model to evaluate before a recommendation is generated. When a class of decisions is defined and the DCA is implemented, we can inexpensively obtain an individual decision analysis considering the specific situation of decision maker within the class.

In this paper, a supervised learning approach is utilized for analyzing a class of decisions which would result in generating IDs. While there are many other supervised learning approaches in neural and non-neural nets that we could have used [Quinlan, 1992; Weiss & Kulikowski, 1991], we choose to use a feed-forward neural net (FNN). Some good reasons for using FNN are, for instance: (1) easy to implement and use by the novice, (2) popular in the literature and we wished to explore their capabilities, (3) software is readily available, etc. However, there are no particular properties of the domain that led us to believe beforehand that neural nets would be superior.

We have not experimented with other approaches, as the main intent of our study was to demonstrate supervised learning, and FNN is a good representative method.

Some studies on constructing IDs [or belief networks] from database can be found in Cooper & Herskovits [1992] and Herskovits & Cooper [1991], and a general method to address this topic is presented by Goodman & Smyth [1993]. Note that these methods have some differences from this paper. For instance, while the aim of previous approaches is the automated discovery of probabilistic dependencies when the nodes in the belief net are given, this paper focuses on (1) search for the nodes in the ID when the specific situation of the decision maker is given, and (2) identify the relationships among the nodes. Additionally, the measure of the relationships is not probability with sum-to-unity.

Ⅱ. INFLUENCE DIAGRAMS

Influence diagrams (IDs) developed as a model for representing complex decision problems based upon incomplete and uncertain information from a variety of sources [Owen, 1984]. Knowledge of the interrelationships among variables is represented in a compact

graphical and numerical framework which identifies the critical variables and explicitly reveals any conditional independence among them. The application of IDs as a development tool for building expert systems or intelligent decision systems has been introduced by Rege & Agogino [1988] and Holtzman [1989]. Kim [1991], Kim *et. al.* [1992] and Chung *et. al.* [1992] have proposed a method of building IDs using rule-based technique for an efficient problem solving.

As shown in Figure 1, ID is an acyclic digraph $G = (N, A)$, where N is a finite set of nodes and A is a set of arcs, $A \quad NN$. This visual level of the ID explicitly reveals the flow of information, influences, and overall structure of the decision problem. The nodes are partitioned into sets C, D and V. The circular-shaped chance nodes cC represent uncertain or certain states, the rectangular-shaped decision nodes dD reveal variables whose values are chosen by the decision maker, and the diamond-shaped value node vV represents the objective to be maximized in expectation by the decision analysis.

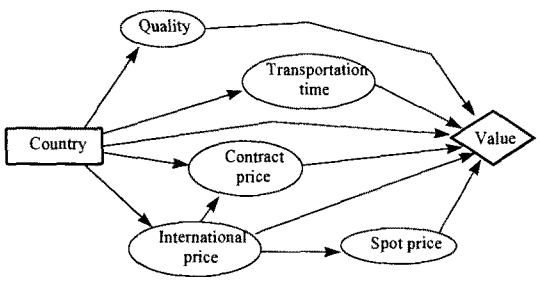


Fig. 1. An example of influence diagram.

As shown in Figure 2, there are five types in A, and each arc in the graph has different meaning. Arcs between cC represent *conditional influences.*

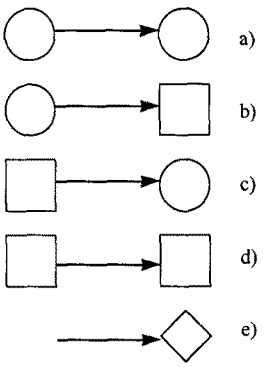


Fig. 2. Five interpretations of an arc in influence diagrams. (a) Probabilistic, (b) Informational, (c) Causality, (d) No-forgetting, (e) Value influence

The absence of this arc is a stronger statement, indicating explicitly the conditional independence between the two variables. An arc going into dD indicates *informational influence* and shows which variable will be known by the decision maker at the time the decision is made. Arcs from d to c node are causalities since the choice of one decision alternative over the other selections can influence the universe of values the state variable can assume. *No-forgetting* arcs are placed between d to signify that decisions are sequential in time and the value of past decisions is remembered. Arcs in the single value node vV signify which nodes directly influence the objective function.

ID can be viewed from three levels: topological, functional, and the numerical level [Howard & Matheson, 1984]. Each level would provide a stage of decision making in a given domain. At the topological or visual level shown in Figure 1, the nodes in the diagram represent the key variables in the system being modeled and the arcs or arrows identify conditional influences among them. The nature

of these influences is specified at the functional level and further quantified at the numerical level. It should be noted that IDs on the topological level do not need a mathematical or probabilistic basis to justify themselves.

The functional level is concerned with how nodes are related. The relationships described in topological level can be divided into two parts, the *conditional influence* (Figure 2. a, c, e) and the *informational influence* (Figure 2. b, d). Their influences are justified by mathematical or probabilistic representation at the functional level. At the final level, numerical level, probability distributions from prior information, decision values and costs, and the utilities of the decision maker are assessed numerically for each node.

Once a complete ID is generated, the diagram is manipulated and evaluated for determining the optimal decision strategy. Probabilistic IDs can be solved using an algorithm described by Shachter [1986, 1988]. This algorithm consists of the value-preserving transformations, node removal and arc reversals, which correspond to the rollback procedure in decision tree models [Bunn, 1984].

III. DECISION CLASS ANALYSIS

3.1. The Concept

Model building of decision domain is known to be a complicated and burdensome process, and then the resulting diagram becomes applicable to only one specific problem. As a result, researchers have investigated the use of knowledge from one decision problem to solve other similar problems. Holtzmen [1989] describes decision class analysis (DCA) which regards a decision analysis as an integrator of decision knowledge and treats a set of decisions having some degree of similarity as a single unit.

Although a concrete example or definition of *similarity* is not found, DCA concept would be helpful in modeling a decision problem in an efficient way. In this research, similarity among decision problems are interpreted in such a way that the frames of influence diagrams (IDs) for each decision problem are mutually resembled, i.e., key variables that should be considered in the problems are partially similar. More specific descriptions of similarity is needed in further research.

Whereas the end result of an individual decision analysis is a decision, the result of a DCA is an individual decision analysis. In contrast with the single decision, DCA implies a deliberate omission of knowledge pertaining to the decision situation. Thus, analyzing a class of decisions occurs at a higher level of abstraction than analyzing a single decision. Figure 3 depicts the knowledge about DCA.

In the construction of a decision model, typically, domain experts provide domain-specific knowledge, while a decision maker furnishes situation-specific knowledge, i.e., information about his or her current situation environment. Domain-specific and situation-specific knowledge is required to efficiently yield a formal model from which a recommendation can be inferred. In its integrator role, decision analysis focuses on acquiring knowledge from the decision participants, thus facilitating the development of a consistent decision model that properly

represents the domain and situation of the decision problem. The ID described previously is a good decision model to characterize the integrator role because it provides an effective communication language between the remote knowledge sources.

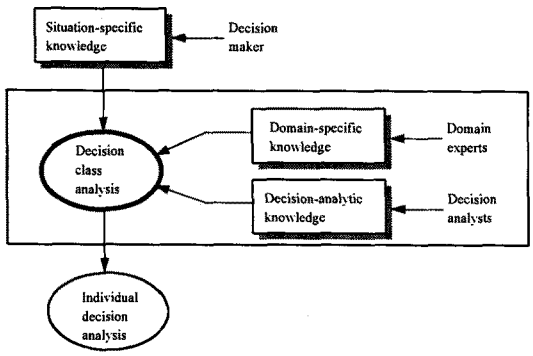


Fig. 3. The knowledge about decision class analysis.

3.2. Influence Diagram and Neural Net in Decision Class Analysis

A major advantages of IDs is the ability to support problem partitioning, decomposition and abstraction. For instance, excluding irrelevant information from the ID can save a decision maker's time and efforts, since there are fewer variables to be interpreted. Generally, variables in the ID are changeable from the current *specific situations*. The specific situations consist of decision nodes and decision maker's circumstances that exclude numerical and functional information, which are called *situation frames* in subsequent descriptions. It is noticed situation frames except decision nodes are not contained in the ID.

Usually recent approaches for implementing DCA use knowledge-based or rule-based approaches [Holtzman, 1989; Kim, 1991; Kim *et. al.*, 1992; Chung *et. al.*, 1992]. Rule-based approaches tend to be domain-specific and function extremely well when decision problems are well defined. Rule-based system implementation can be a lengthy process depending on the size of the domain and the range of cases. Namely, these are implemented by representing the domain expert's knowledge as a series of IF-THEN conditions, which depends on observation of the whole known combinations of the expert's data. If the number of data points is large and/or associative, then the rule-based approaches may be inappropriate.

As described earlier, DCA treats a set of individual decisions and it occurs at a high level of abstraction. That is, a class problem consists of a number of individual decision problems, thus the size of class problem is usually larger than the size of each individual decision. When given the situation-specific information from the decision maker, the DCA should abstract the corresponding specific decision variables for solving the individual problem. In this case, the DCA can be described as a *classification problem.*

In contrast to rule-based systems, neural networks have a broad response capability because of their capability to provide the general classification of a set of inputs [Zahedi, 1991]. They can capture a large number of cases quickly and provide reasonably accurate responses. More specifically on the perspective of implementing a DCA, consider the number of the possible subsets of the situation frame set in a class problem. That is $2n$-1, where n is the total number of the elements of the situation frame set (the reason of -1 is to exclude the case the number of subset elements is zero). For n=9, the number of possible cases

is 511. Although it is possible to generate the 511 numbers of rules, it is an inefficient way which requires much time and effort. Thus, neural network with generalization capability would be an excellent way of implementing the DCA.

IV. IMPLEMENTING DECISION CLASS ANALYSIS

4.1. A Procedure to Build a Decision Model

Analyzing a class of decisions is composed of three steps. First, the decision maker decides decision nodes to represent the decision-making purpose of a given problem. Second, the decision maker suggests knowledge of specific situations which occurred at current circumstance and situation. The well-represented situation-specific knowledge plays a major role to elicit a single decision analysis through the DCA. In the third step, to obtain a single decision analysis, an influence diagram (ID) is built based on the decision and the situation-specific knowledge. The third step is made of two phases: Phase I is to search for relevant chance and value nodes of the individual ID from the given decision and specific situations. Phase II elicits arcs among the nodes.

The first two steps are performed by the decision maker. The third step is a major part of DCA, which is done by two trained feed-forward neural nets (FNNs): Neural Net I for Phase I and Neural Net II for Phase II. To implement the third step, the training set is obtained by collecting the knowledge of decision participants in a class of decision problems. The overall procedure is depicted in Figure 4.

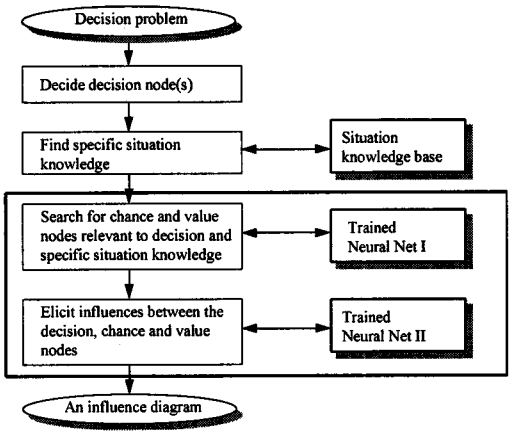


Fig. 4. Overall procedure for implementing the decision class analysis.

Neural Nets I and II are trained by the training set using backpropagation learning algorithm. When the training is completed, the trained neural nets perform DCA, i.e., the ID is generated from the decision and situation-specific knowledge. Therefore, two trained neural nets have to perform two phases respectively, which are explained in detail in subsequent sections.

4.2. Building Influence Diagram

Phase I is to search for relevant chance and value nodes based on given decision and situation frames. The situation frame described in the previous section also implies an individual situation-specific knowledge. Phase II elicits influences among decision, chance and value nodes.

Denote a set S, where there may be one or more situation frames sS. For each node in the

ID (*dD*, *cC* and *vV*) or situation frame, if it is present then its value is 1, otherwise its value is zero. The arcs *aij* have three values: if the direction of the arc is (i,j) then *aij* =1, if (j,i) then *aij* = -1, and if the influence between i and j does not exist then *aij* = 0, where $i < j$ and i and j are nodes indices. Then *DS*, *CV*, *DCV* and *ARC* are denoted as follows:

$$DS=\{d_i \mid i=1, P\}\bigcup\{S_j \mid j=1, n\},$$
$$CV=\{c_i \mid i=1, q\}\bigcup\{v_j \mid j=1, m\},$$
$$DCV=\{d_i \mid i=1, P\}\bigcup CV,$$
$$ARC=\{a_{ij} \mid i<j,\ i=1, p+q+m-1, j=2, q+p+m\},$$

where the size of *ARC* is $\binom{q+p+m}{2}$

Phase I

So as to search for relevant chance and value nodes from the given decision and situation frames, Neural Net I is used to represent the relation between chance and value nodes, and decision nodes and situation frames. The training set of input and desired output pairs to learn Neural Net I is represented as . These training pairs can be generated from the case studies of the similar problems within the class with the help of the decision maker and domain expert.

Phase II

A trained Neural Net II has to be prepared to elicit the influences among output nodes of Phase I. The training set of input and desired output to learn Neural Net II is . Namely, the desired output of Phase I is the input of Phase II and the desired output of Phase II is the arcs of an ID. Hence the training pairs can be generated consecutively based on those of Phase I.

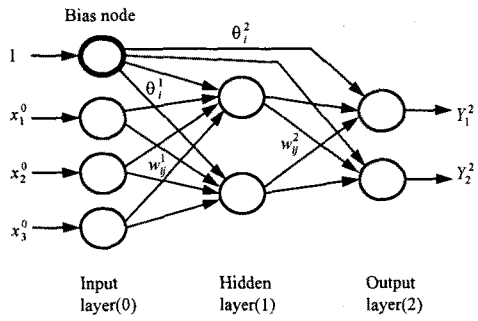


Fig. 5. A two-layer feedforward neural network.

As described in Phases I and II, this study requires supervised learning because the pair of each input vector with a target vector representing the desired output exists explicitly. There is a tremendous variety of training algorithms in use today. The backpropagation algorithm is a systematic method for training multi-layered FNNs, and it has a strong mathematical foundation [Rumelhart & McClelland, 1986]. Although any type of neural nets may be used, this paper uses a FNN shown in Figure 5 because one may use it as one of the most simple model.

V. SIMULATION WITH AN EXAMPLE

5.1. Description of a ClassProblem

As a class of decision problem, a raw-material buyer problem in a textile cooperation is used to demonstrate building an ID or implementing the DCA. The company makes some types of synthetic products. The main raw materials of these products are TPA

(terephthalic acid), DMT (dimethy terephthalate) and EG (ethylene glycol). Presently, the raw materials are imported from the foreign market. There are two kinds of decisions in this problem. The manager of the materials section has to decide the amount of raw material and the country from which the goods are imported.

The state related with the decisions is varied with the types of goods and the affecting specific situations. According to a given raw material and the specific situation, variables affecting the decision made a difference. For example, under the situation to buy EG and to decide the country, variables (chance nodes) affecting this decision may include contract price, international price, quality and transportation time. In the case of EG, quality and transportation time are regarded as especially important ones compared with TPA and DMT. On the other hand, in the situation to buy TPA and to decide the contract amount and country, chance nodes related with this decision are inventory level, product demand, contract price, international price and reliability (in the case of TPA, reliability is regarded as an important variable).

In this problem, decisions of a similar type are to be made frequently and these decisions strongly depend on the specific situations (situation frames). Hence, once the DCA is implemented using neural nets, we can greatly reduce the time and save large amounts of duplicate efforts for a single decision. Using the notations described in the previous section, we restrict the boundary of the raw-material buyer problem in Table 1.

Table 1. Summary of the raw-material buyer problem.

Node name	Symbol	Content
Decision	d_1	Contract amount
	d_2	Country
Chance	c_3	Product demand
	c_4	Inventory level
	c_5	International price
	c_6	Spot price
	c_7	Contract price
	c_8	Reliability
	c_9	Transportation time
	c_{10}	Quality
Value	v_{11}	Value
Situation	s_1	Buying TPA raw-material
	s_2	Buying DMT raw-material
	s_3	Buying EG raw-material
	s_4	Variation of OPEC policy
	s_5	Variation of domestic economy
	s_6	Variation of foreign economy
	s_7	Variation of opposite company policy

5.2. Learning Procedures

The backpropagation algorithm does not always find global minimum but may stop at a local minimum. However, in most cases, the system can usually be driven to the global minimum or to the desired accuracy with an appropriate choice of the number of hidden layers and processing elements (PEs). There are no general rules in the literature that defines the number of hidden layers, and the number of PEs per hidden layer. Most studies of backpropagation algorithm have found no more than two hidden layers are required [Wasserman, 1989; Hecht-Nielsen, 1989]. The classification problem of this paper does not possess linearly separable classes, thus a two-layer neural net with one hidden layer is used [for this issue, see Lippmann, 1987].

The addition of more hidden PEs improves the performance of training but the ability of generalization gets worse [Sorsa *et. al.*, 1991]. The number of hidden PEs must be large enough to form a decision region that is as complex as required by the given problem, and on the other hand is small enough that the generalization ability remains good. A study similar to this statement is formally conducted by Geman *et. al.*, [1992]. They present error in training is partitioned into bias and variance: the more the number of hidden PEs are used, the larger the variance but the smaller the bias is occurred. Hence, we start with a small number of hidden PEs and increase the number until it becomes possible to drive the learning error to a desired accuracy.

Initial weights and thresholds are used as random numbers within a small range [-0.3, 0.3], although any other range, for example, [-0.2, 0.2], could be used. The learning coefficients of the generalized delta rule are selected such that all of the step sizes with respect to weights and thresholds are 0.9 and all of the momentum term are 0.6 because they are reported to yield fast learning [Rumelhart & McClelland, 1986].

Training for Phase I

In the raw-material buyer problem, the number of input PEs is 9 (the number of situation frames) and the number of output PEs is 11 (the number of *DCV* elements). All PEs in Neural Net I are transferred by the sigmoid functions, although any transfer function could be used. About 50 data pairs have been collected with the help of decision makers and domain experts. Among the data pairs, 30 samples are used for training and the remainders are used to test the learning performance.

The number of hidden PEs of Neural Net I is increased from a low number up to a level that drives the actual output to a desired accuracy. Specifically, if the node of ID is presented as 1, Neural Net I should produce the output near 1. In the other case it should produce the output near zero. Namely, the output of Neural Net I has to be clearly discriminated whether the node is present or absent. For example, the output can be interpreted as a node absent if an output is smaller than 0.5 and a node present if larger than 0.5. The other discrimination criteria could be used such as 0.4 and 0.6, or more strongly 0.3 and 0.7, etc.

Training for Phase II

The number of input PEs equals the number of output PEs in Phase I (i.e., 11). The possible maximum number of output PEs (i.e., arcs) of the raw-material buyer problem is 55. Among them, the arcs that are not always present are eliminated for the sake of simple training. For example, $a_{3,7}$ is always zero because of no influence between product demand (c_3) and international price (c_7). Consequently 23 output PEs are actually used.

Instead of using the sigmoid functions, hyperbolic tangent functions may be used in this phase to transfer the PEs of Neural Net II, because they can range from -1 to 1. The hyperbolic tangent functions make it possible to express the directions of the arcs: forward direction as 1, reverse direction as -1, and no influence as 0. The method of choosing the number of hidden PEs in Phase I also applies to Phase II.

Table 2. Discriminations by increasing hidden PEs for Phase I.

No. of hidden	PEs	2	3	4	5	6	7
Criteria of discriminations	training set	no	no	0.66 0.39	0.71 0.21	0.73 0.20	0.81 0.19
	testing set	no	no	no	0.66 0.33	0.68 0.33	0.78 0.24

5.3. Discussion of Simulation Results

Results of Phase I

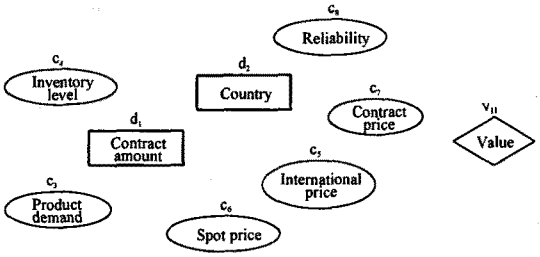
Initially, the 9-2-11 Neural Net I is used. As shown in Table 2, the addition of more hidden PEs improves the discrimination power. In the case of 4 hidden PEs, the actual output of Neural Net I is discriminated such that the value smaller than 0.39 represents the node absent and the value larger than 0.66 the node present. However, with the output in the testing set, we cannot discriminate whether the node is present or not. In the case of 5 hidden PEs, the value smaller than 0.33 is interpreted as the node absent and the value larger than 0.66 as the node present. Consequently, in this phase 5 hidden PEs are needed at the least.

Table 3. Discriminations by increasing hidden PEs for Phase II.

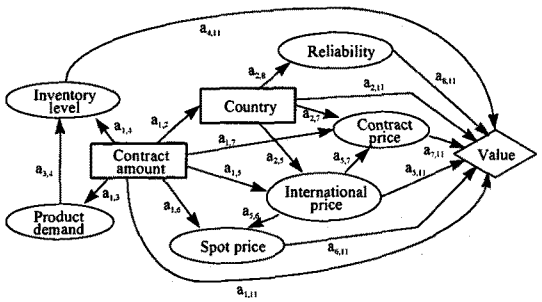
No. of hidden	PEs	3	4	5	6	7	8
Criteria of discriminations	training set	no	0.67 0.29	0.75 0.25	0.79 0.21	0.82 0.19	0.83 0.16
	testing set	no	0.60 0.41	0.61 0.32	0.70 0.32	0.73 0.29	0.74 0.24

Results of Phase II

The 11-5-23 Neural Net II is used initially. Table 3 shows the addition of more hidden PEs improves the discrimination power. Observe that the present and absent of arcs can be discriminated when 4 or more PEs are used. Shown in Figure 6 is an example generated by Phases I and II.



(a) Output of Phase I.



(b) Output of Phase II.

Fig. 6. An influence diagram built by the two phases.

Results by the Modification of Training Set Size

Most important question is how many training data is used for solving a given classification problem. There is no general rules about this question and the rule may be different according to given problem. The number of IDs (i.e., classes) that mainly or frequently used in raw-material buyer problem is about 50-60. Among them, 30 training pairs (50-60%) are used for training Neural Nets I and II, which are randomly selected over the classes. Then, we can obtain a satisfying solution when more than 5 hidden PEs are

used in Neural Nets I and II. Shown in Figure 7 is a result obtained by reducing the number of training pairs from 30, where error is the number of ambiguous results (i.e., non-discriminated data whether nodes in the ID are present or not). It gives that when less than 30 data pairs are used for training Neural Nets I and II, there is no way to solve the classification of raw-material buyer problem although any number of hidden PEs are used.

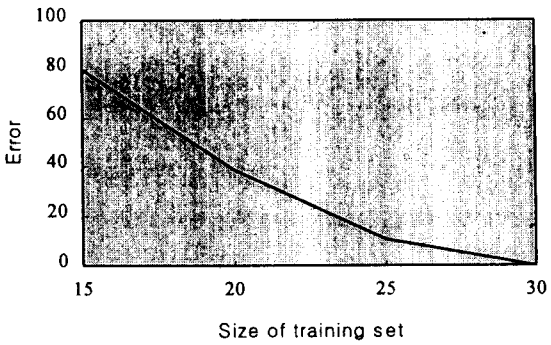


Fig. 7. Error according to the size of training data.

VI. CONCLUDING REMARKS

Analyzing a class of decisions as a single unit is a basic concept to model a decision problem efficiently. This study proposes a FNN approach to implement the DCA, and as a result, to build an ID. With the trained Neural Nets I and II, the decision maker can easily obtain the ID of a specific decision problem.

The illustrative example used in this paper may be a small class of decisions. The simulation results in each phase are quite satisfactory. However, there are some lacks in this paper: for instance, does not clearly present a description of training set that required according to increasing the size of a class decision problem, and the definition of DCA. The lack makes it a promising area for further research.

⟨REFERENCES⟩

Bunn, D.W. [1984] *Applied Decision Analysis,* New York: McGraw-Hill, 1984.

Chung, T.Y., Kim, J.K. and Kim, S.H. [1992] "Building an influence diagram in a knowledge-based decision system," *Expert Systems With Applications,* 4, No. 1, 1992, pp.33-44.

Cooper, G.F. and Herskovits, E. [1992] "A bayesian method for the induction of probabilistic networks from data," *Machine Learning,* 9, 1992, pp.309-347.

Geman, S., Bienenstock, E. and Doursat, R. [1992] "Neural networks and the bias/variance dilemma," *Neural Computation,* 4, 1992, pp.1-58.

Goodman, R.M. and Smyth, P. [1993] "Automated induction of rule-based neural networks from databases," *International Journal of Intelligent Systems in Accounting, Finance and Management,* 2, 1993, pp.41-54.

Hecht-Nielsen, R. [1989] *Neuralcomputing,* Reading, MA: Addison-Wesley, 1989.

Herskovits, E. and Cooper, G.F. [1991] "Kutato: entropy-driven system for construction of probabilistic expert systems from databases," in Bonissone, P.P., Henrion, M., Kanal, L.N. and

Lemmer, J.F. (eds), *Uncertainty in Artificial Intelligence*, Vol. 6, 1991, pp.117-125.

Holtzman, S. [1989] *Intelligent Decision Systems*, Addison-Wesley, 1989.

Howard, R.A. and Matheson, J. [1984] "Influence diagrams," in Howard, R.A. and Matheson, J.E. (eds), *Readings on the Principles and Applications of Decision Analysis, Vol. II*, Menlo Park, CA, Strategic Decision Group, 1984.

Kim, J.K. [1991] *A Knowledge-Based System for Decision Analysis*, Ph.D. thesis, Department of Industrial Engineering, KAIST, Korea, 1991.

Kim, J.K., Chung, T.Y. and Kim, S.H. [1992] "KIDS: a knowledge-based decision system to build an influence diagram," *A First World Congress on Expert Systems*, Miami, Florida, 1992.

Lippmann, R.P. [1987] "An introduction to computing with neural nets," *IEEE ASSP Magazine*, 4, 1997, pp.4-22.

Owen, D.L. [1984] "The use of influence diagrams in structuring complex decision problems," in Howard, R.A. and Matheson, J.E. (eds), *Readings on the Principles and Applications of Decision Analysis*, Vol. II, Menlo Park, CA, Strategic Decision Group, 1984.

Quinlan, J.R. [1992] *Programs for Machine Learning*, Morgan-Kaufmann, 1992.

Reed, J. [1989] "Building decision models that modify decision systems," *Proceedings of the Thirteenth Annual Symposium on Computer Applications in Medical Care, 1989.*

Rege, A. and Agogino, A.M. [1988] "Topological framework for representing and solving probabilistic inference problems in expert systems," *IEEE Transactions on System, Man and Cybernetics*, 18, No. 3, 1988, pp.402-414.

Rumelhart, D. and McClelland, J. [1986] *Parallel Distributed Processing*, Vol. 1, Cambridge: MIT Press, 1986.

Shachter, R.D. [1986] "Evaluating influence diagrams," *Operations Research*, 34, 1986, pp.871-882.

Shachter, R.D. [1988] "Probabilistic inference and influence diagrams," *Operations Research*, 36, 1988, pp.589-604.

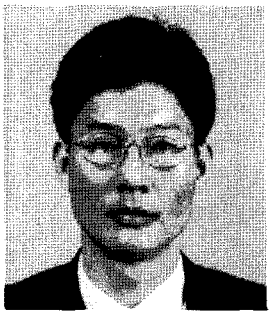
Sorsa, T., Koivo, H. and Koivisto, H. [1991] "Neural networks in process fault diagnosis," *IEEE Transactions on System, Man and Cybernetics*, 21, No. 4, 1991, pp.815-824.

Wasserman, P.D. [1989] Neural Computing: *Theory and Practice*, New York: Van Nostrand Reinhold, 1989.

Weiss, S.M. and Kulikowski, C.A. [1991] *Computer Systems That Learn*, Morgan-Kaufmann, 1991.

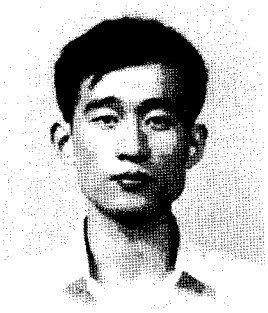
Zahedi, F. [1991] "An introduction to neural networks and a comparison with artificial intelligence and expert systems," *Interfaces*, 21, No. 2, 1991, pp.25-38.

◆ 저자소개 ◆



박경삼
부산대학교 산업공학과(1989)를 졸업했으며, 한국과학기술원 산업공학과에서 석사(1991) 및 박사학위(1996)를 취득하였다. 한국과학기술원 경영정보공학과에서 Post-Doc. 과정을 거쳐 현재 미국 텍사스주립대학 (Austin) 경영과학과(MSIS)에서 초빙연구원 (Visiting Scholar)으로 재직중이다. 관심분야로는 Multi Obj. Optimization, MCDM, Mathematical Programming, DSS/ES 등이다.

김재경
현재 경기대학교 경영정보학과 조교수로 재직중이다. 서울대학교 산업공학과(1985)를 졸업하고 한국과학기술원 산업공학과에서 석사(1987) 및 박사학위(1991)를 취득하였다. 대전산업대학교 산업공학과에서 4년간 재직하였으며, 미국 Minnesota 대학교 경영정보학과에서 1년간 초빙교수를 역임하였다. 주요 관심 분야로는 DSS/ES, DataBase, BPR, 정보시스템 평가 등이다.



윤형재
현재 LG EDS Consulting부문 ERP팀 과장으로 재직중이다. 1990년에 부산대학교 산업공학과를 졸업하고, 1992년과 1996년에 한국과학기술원 산업공학과에서 각각 석사 및 박사학위를 취득하였다. 주요 관심 분야로는 품질관리, 신뢰성공학, CIM, MIS 등이다.