Int J Adv Manuf Technol (2008) 38:1061–1074
DOI 10.1007/s00170-007-1152-3

ORIGINAL ARTICLE

Identification of product definition patterns in mass customization using a learning-based hybrid approach

Li Yu · Liya Wang · Jianbo Yu

Received: 19 December 2006 / Accepted: 4 July 2007 / Published online: 11 August 2007

Abstract Mass customization, which aims at satisfying individual customer needs with near mass production efficiency, has become a major trend in industry. Adopting the mass customization paradigm, customer preferences have a significant impact on the product design process. Thus, it is important for companies to make proper decisions in translating the voice of customers to product specifications. To facilitate this process, a learning-based hybrid method named KBANN-DT is proposed, which combines knowledge-based artificial neural network (KBANN) and CART decision tree (DT). In this method, the KBANN algorithm is applied to modeling the relationship between customer needs and product specifications. With prior domain theory, KBANN can provide a high generalization performance even if the data set is small. Based on the trained KBANN network, the CART DT algorithm is employed to extract rules from it. To illustrate the effectiveness of the proposed method, a case study in an elevator company is reported. The results show that the proposed method can be a promising tool for product definition.

Keywords Customer needs · Decision tree · Functional requirements · Knowledge-based artificial neural network · Mass customization · Product definition

1 Introduction

In the current scenario of globalization and strong competitive environment, it has become imperative for companies to satisfy a wide spectrum of customer demands within limited products. To meet this challenge, companies pursue a strategy of mass customization (MC), which makes the identification and fulfillment of the wants and needs of individual customers without sacrificing efficiency, effectiveness, and low costs [1]. Accordingly, understanding what customers really want is at the center of overall design process for mass customization. However, customer needs (CNs) are normally qualitative and tend to be imprecise and ambiguous owing to their linguistic origin [2]. Furthermore, customers and designers often hold different views to a same product, which often leads to misunderstanding between them. Therefore, it is demanding for a manufacturer to uncover customer preferences to be more competitive in the ever-changing market.

Product definition (PD), to capture and understand customer needs effectively and subsequently to transfer them into design specifications, is one of the essential premises for successful product design in today's competitive global market [3]. During this phase, customers, marketers and engineers work together to make proper decisions in translating the voice of customers into product specifications (e.g., specific functional requirements). In a mass customization system, enterprises need to satisfy diverse niches while maintaining near-mass production efficiency [4]. To improve the commonality and modularity, certain customer needs are often mapped into a "right" product family (PF) and then detailed functional requirements (FRs) are derived from a functional perspective of product family architecture (PFA) [5, 6]. Hence, product definition involves two main stages. The first stage is to identify product definition patterns. In this stage, the mapping relationship between customer needs and product families is investigated. The second one is called the "refinement" process [7]. The key issue of this stage is to

L. Yu (✉) · L. Wang · J. Yu
Department of Industrial Engineering and Management,
Shanghai Jiao Tong University,
Shanghai 200240, People's Republic of China
e-mail: yuli_sjtu@163.com

transform specific customer needs into product specifications, which is a "zigzag" process [8] between customer domain and functional domain.

Being the front end of the whole design process, decisions made in the first stage of PD have great influence on the cost and customer satisfaction. Although it is the key to a successful design, this stage receives only limited attention. During this stage, two major goals are pursued. *The first is the quality of decision*: modeling the mapping relationship between CNs and PFs with high accuracy. *The second is the reason of decision*: provide engineers with concise knowledge of customer preferences. This issue is as important as, if not more important than the first one. Nevertheless, these two goals are not easy to be fulfilled for the following reasons:

(1) Customer needs are often fuzzy and imprecise;
(2) The mapping relationship between customer domain and functional domain is often complicated nonlinear;
(3) Design knowledge of product definition incompletely and implicitly exists in experts and historical database.

Various approaches and strategies for product definition have been reported in literature. Jiao and Chen reviewed research work carried out toward customer requirements elicitation, analysis, and specification [9]. Quality function deployment (QFD) is widely used in the early phase of requirements and specifications, which links both marketing and engineering [10, 11]. By means of house of quality, QFD has become a way to translate the voice of customers into design specifications. To capture the voice of customers, Chen [12] used a laddering technique in the elicitation of customer requirements and used the ART2 neural network in evaluation and market analysis. With adaptive conjoint analysis, Du et al. proposed a systematic approach to understand customer needs in product customization [13]. Other analytical techniques, such as Kansei engineering [14], sorting technique [15], are useful for making rational decisions in product design. However, these methods assume that the product development process starts from "a clean sheet of paper", which may result in long lead time and uncontrollable variety [5].

To take advantage of the wealth of customer requirement information accumulated in existing products, many researches investigate the mapping relationship between customer needs and functional requirements with machine leaning techniques [2]. On the basis of functional requirements classification, Jiao et al. [8] proposed a classification tree algorithm to realize the mapping in product definition. However, the classification tree algorithm is sensitive to the splitting rule and the structure of the tree may be not be appropriate or not very predictive. Chen et al. used neural networks (NNs) to facilitate product definition in mass customization [16]. Although NNs are capable of learning complex nonlinear relationships and perform well with missing or incomplete data [17], its architecture is viewed as a black-box that is difficult to understand. To discover knowledge from past sales and product records, Jiao et al. [18] tried to find the relationship between customer needs and product specifications with association rule mining techniques. Related work by Shao et al. [19] used data mining and rough set to tackle the relationship between customer needs, product specifications, and configuration alternatives.

Most research mentioned above investigated product definition based on either domain theory (a set of inference rules with respect to customer preferences) or empirical cases (historical database). However, it is a common scene that both knowledge resources incompletely exist in a company and have intersection or not. It is obvious that the union of them is bigger than any single one. Thus, eliciting design knowledge from both information resources, engineers can better understand the relationship between customer needs and design requirements.

In this paper, we describe a learning-based hybrid method named KBANN-DT (knowledge-based artificial neural network-decision tree) to facilitate the first stage of product definition-identifying product definition patterns. Integrating the merits of KBANN and classification tree, this mechanism learns from the two knowledge resources. With its help, designers can work out specific products by referring to previous successful solutions instead of attempting a completely new design. It is expected to shorten the lead time of PD, especially in the current knowledge-intensive environment.

This method can be divided into two steps. First, knowledge-based artificial neural network (KBANN) [20] is applied to modeling the mapping relationship in PD with two knowledge resources, i.e., domain theory and empirical cases. In this algorithm, the initial topology of the neural network is determined by domain theory, which is then refined through training. Many researches [21, 22] demonstrates that KBANN generalizes better than the algorithms that only utilize one of these information sources, such as standard backpropagation, decision tree, EITHER, etc. It can perform well when examples and domain theory are insufficiently available. Table 1 shows the summarized reasons of adopting the KBANN algorithm for product definition. Then, with this algorithm, both information resources that incompletely exist in a company can be utilized to increase the quality of decisions. However, a major subject of the criticisms is the unexplainable nature of a trained KBANN network. Then, in the second step, refined and comprehensible rules are needed to be extracted from KBANN. Instead of using the MofN method proposed by Towell [23], we use CART decision tree [24] to extract rules as it is easier to understand and convert to IF...THEN rules. It is similar to the ANN-DT algorithm proposed by

Table 1 The rationale for adopting the KBANN algorithm for product definition

Product definition characteristic	Corresponding reason for adopting KBANN
Cannot be represented by simple mathematical formula	Can be used to modeling linear or nonlinear complex problems
Contains prior expert knowledge and historical database	Combines symbolic and empirical learning techniques
Identifies customer preferences correctly	Can achieve high accuracy
Holds prior expert knowledge that may be partially correct	Is robust to domain theory
Holds limited historical cases	Requires less data

Schmitz et al., which is able to extract rules from a neural network [25, 26]. As the ANN-DT algorithm does not depend on any assumptions with regard to the structure of the neural network [25], it can be used to understand the behavior of the trained KBANN network by means of heuristic rules. With the extracted rules, design engineers can better understand customer preferences.

The rest of this paper is organized as follows. In Sect. 2, the framework of product definition based on KBANN-DT is discussed. In Sect. 3, the construction of KBANN-DT is illustrated. In Sect. 4, a case study of identifying product definition patterns in an elevator design is presented with the focus on its learning and interpreting ability. Finally in Sect. 5, the main contribution of this paper is summarized and future work is discussed.

2 Problem formulation

This research addresses the decision problems in the early stage of product design, which begin with customer needs and end with a set of functional requirements. It involves zigzagging mapping process between customer domain and functional domain with creative conceptualization [8]. In practice, most new products evolve from existing products, i.e., so-called variant design [18]. Following this opinion, Fig. 1 illustrates the principle of product definition based on the KBANN-DT method. In a company that pursues MC strategy, FR patterns (including FR topology, FR classification and FR templates) act as the base products to be modified to meet specific customer needs [5]. In this way, manufactures can make use of historical data, design theory, product evolution paths, and feedback from customers on current products. It maintains the integrity of the product family and the continuity of the infrastructure, thereby leveraging existing design and manufacturing investments [2].

In the customer domain, customer needs can be described as a set of features or attributes. Generally, Parato analysis or factor analysis is used to identify the customer requirement variables $\{\mathbf{CV}_p\}$, $\mathbf{CV}_p=(cn_1, cn_2,..., cn_I)$, $p\in\{1, 2,..., P\}$ where P is the total number of customers and I is the number of customer features or attributes. Each feature, $cn_i \mid \forall i\in[1, 2,..., I]$, may take on one out of a finite set of options, $cn_i \in \{cn_i^1, cn_i^2, \ldots, cn_i^{J_i}\}$, where J_i denotes the option number of the i*th* feature. Thus the requirement information of a particular customer, $\mathbf{CV}_p$, can be depicted by a vector of certain options of these features, for example, $\mathbf{CV}_p=(cn_1^2, cn_2^1,..., cn_I^3)$, where cn_1^2 refers to the second option of feature cn_1 that is chosen by the p*th* customer, cn_1^2 the first option of feature cn_2, and cn_I^3 the third option of feature cn_I. The customer domain is then characterized by a set of vectors, $\mathbf{\Omega_{CV}}=\{\mathbf{CV}_1, \mathbf{CV}_2,..., \mathbf{CV}_p\}$.

From the functional perspective, a product family contains a set of functional requirements and their topology. It can be represented by a representative center vector and variation ranges, i.e., FR template (**FRT**) [5, 8]. Usually, fuzzy clustering analysis technique is used to cluster similar functional requirements [18]. Then, a product family is noted as a tuple: $\mathbf{FRT}k=\mu_k, \Delta_k, k\in\{1, 2,..., K\}$, where K is the total number of FR templates, μ_k is the center vector of the k*th* FR template, and Δk is the variation range of the vector μ_k. Here, $\mu_k=(\mu_k^1, \mu_k^2,..., \mu_k^M)$, $\Delta_k=(\delta_k^1, \delta_k^2,..., \delta_k^M)$, where M is the number of product attributes (FRs), $\mu_k^m \mid \forall m\in[1, 2,..., M]$ is the center value of the mth product attribute, and $\delta_k^m \mid \forall m\in[1, 2,..., M]$ is the variation range of the mth product attribute. The functional domain is then characterized by a set of vector tuples $\mathbf{\Omega_{FRT}}=\{\mathbf{FRT}1, \mathbf{FRT}2,..., \mathbf{FRT}k\}$.

Based on the company's sales records and product documentation, customer transaction records (T) is defined as: $\mathbf{T}\sim\{(\mathbf{CV}_q, \mathbf{FRT}_q) \mid q=1, 2,..., Q\}$ where $\mathbf{CV}_q\in\mathbf{\Omega_{CV}}$, $\mathbf{FRT}_q\in\mathbf{\Omega_{FRT}}$ and Q is the number of historical transaction records. The underlining meaning of the coupled data, $\mathbf{CV}_q$ and $\mathbf{FRT}_q$, is the mapping relationship between customer and functional domain. Hence, these transaction data are used for the training in the KBANN-DT algorithm to obtain the internal logic of the design process.

Domain theory, another knowledge resource in a company, plays an important role throughout the whole design process. In this research, domain theory of product definition refers to the inference rules between CN attributes and FR templates. In the form of a set of IF... THEN rules, domain theory embodies the customer preference to different product families. Taking computer design as an example, inference rules can be: *IF* customer requires a computer with high game performance and advanced multimedia *THEN* P4 2.8G series computer is

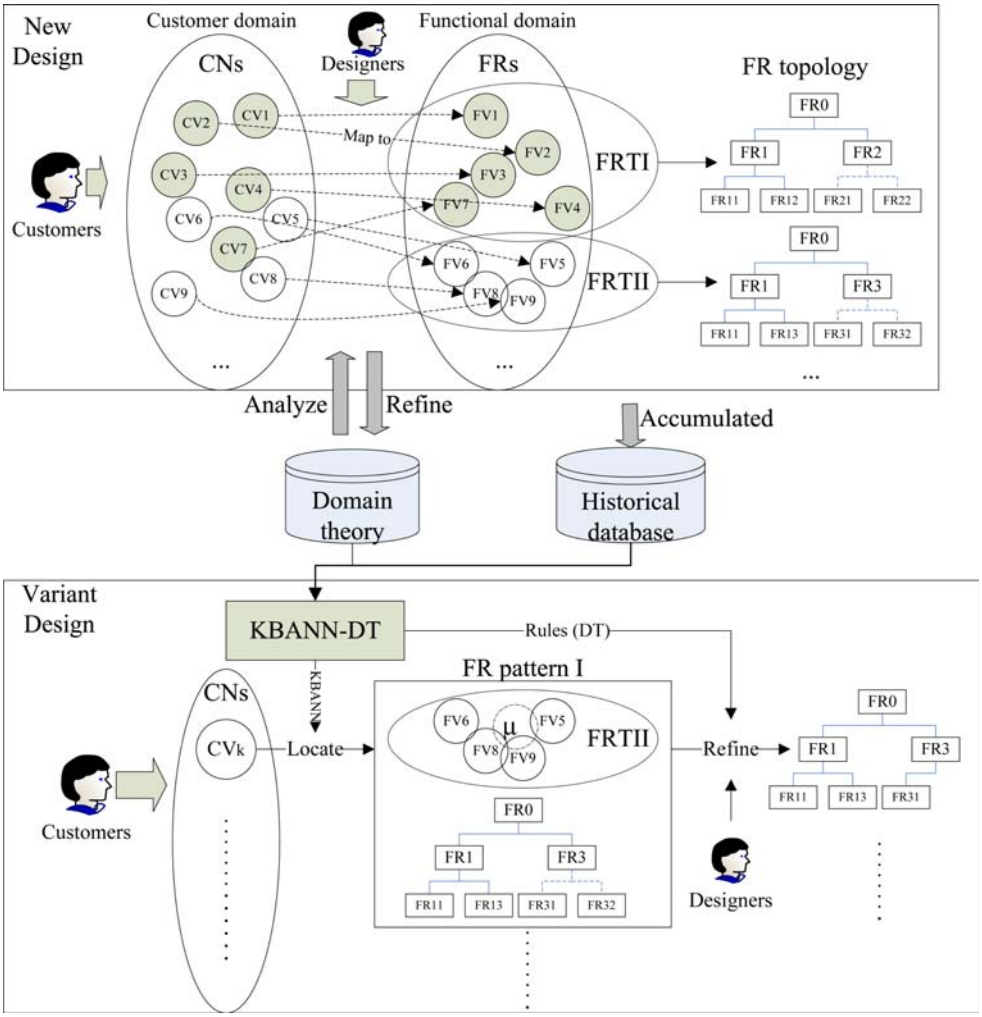


Fig. 1 Product definition based on the KBANN-DT method

preferred; *IF* customer requires a computer with normal game performance and basic multimedia *THEN* P4 1.8G series computer is preferred.

Symbolic rules, the inference rules of PD, can be incomplete or approximately correct. It is used for the construction of a neural network. Hence, nodes distribution and link weights of the network are specified based on the initial knowledge of product definition. Then, more nodes and links can be added for default rules. These newly added units would allow the network to learn relations not anticipated in the pre-existing symbolic rules. At last, the network is trained with historical transaction records. It enhances the performance of the network and makes it reliable for product definition.

These symbolic rules that reflect the mapping relationship in PD can be extracted from design engineers in a company. It is achieved by the following steps. At first, the statements of PD extracted are grouped into different categories based on the CN features they reflected. Similar statements are merged into a single statement. The simplified statements are then transformed into a set of hierarchically structured IF...THEN inference rules required by the symbolic module of KBANN.

Following these points, the proposed method aims to study the mapping relationship in PD from the above two knowledge resources. In general, product definition of variant design is performed by the following procedures.

Firstly, historical transaction records and domain theory are accumulated in a knowledge base.

Secondly, a knowledge-based neural network is constructed. This procedure converts the inference rules into a neural network, which is used as a starting point for incremental learning. The input nodes of the neural network correspond to attributes of customer needs. The hidden nodes are decided by the hierarchical structure of symbolic rules, whereas the output nodes correspond to final conclusions of rules, i.e., product families.

Thirdly, customer transaction records are used as the training set for the knowledge-based neural network. As these historical data embodies the mapping relationship of customer needs and product families, the network is able to identify product definition patterns at a high accuracy. When

a customer presents his requirement for a product, the requirement is fed to the network and designers can locate a suitable product family by the output of the network.

Fourthly, a rule extraction algorithm is performed to understand the behavior of the knowledge-based neural network. It is helpful for understanding the decisions made by the network.

Finally, on the basis of the located product family (coupled FR pattern) and the rules extracted from the trained KBANN network, design engineers are able to work out product specifications with less effort and shorter time. In this way, new product specifications can be developed based on the existing product families to meet various customer requirements, which can improve commonality and modularity of these products.

3 The KBANN-DT method

Identifying product definition patterns is a crucial step because design information is incomplete and design knowledge is minimal at the beginning of product definition. Generally speaking, design knowledge can be divided into two types. One is company's sales records and product documentation, i.e., database of empirical cases; the other is domain theory, from which initial symbolic rules can be obtained. To make full use of the wealth of a company, both should be utilized during this stage.

The general diagram of the KBANN-DT approach is presented in Fig. 2. It is comprised of two steps. The first step is quite similar to the method presented by Towell [20]: the KBANN algorithm. Its structure is constructed with symbolic rules and then is refined by training. After training, KBANN can map different customer needs into corresponding FR templates. However, neural learning in KBANN ignores the symbolic meaning of the initial network [20]. That is, determining exactly why this trained KBANN network makes a particular decision is still a daunting task.

Many researchers have shown that useful rules can be extracted from a trained neural network [27, 28]. In this paper, we use a method of extracting a decision tree from input data generated from a trained neural network [25, 29]. The decision tree algorithm is one of data mining technologies with several advantages:

(1) It executes fast
(2) It can handle a high number of records in a high number of fields with predictable response times
(3) It can be better understood and easily be translated into IF...THEN rules.

For many cases, a decision tree extracted from the input data generated from a trained neural network has a higher generalization ability than doing it from the original training data set [25, 29]. Then, the proposed method is expected to extract a decision tree with higher accuracy, i.e., achieving better rules. These rules can help design engineers grasp the knowledge acquired by KBANN and then make proper decisions to meet various customer requirements. Also, these rules can be used to enrich domain theory that improves the generalization of the KBANN.

Therefore, the method is able to capture the high accuracy of KBANN and the comprehensibility of the decision tree. It is important to emphasize here that this method can not take the place of engineers' creative work. However, it does alleviate the engineer's burden in the refinement process of product definition, which is often a tedious, time-consuming, and error-prone process. The detailed description of this method is discussed as below.

For convenience in demonstrating the KBANN-DT method, the following assumptions are made:

(1) Historical transaction data is stored in database
(2) Domain theory exists in design engineers.

3.1 KBANN

Knowledge-based artificial neural network (KBANN), combining empirical and symbolic learning techniques, is a learning-based hybrid intelligent system that is built on

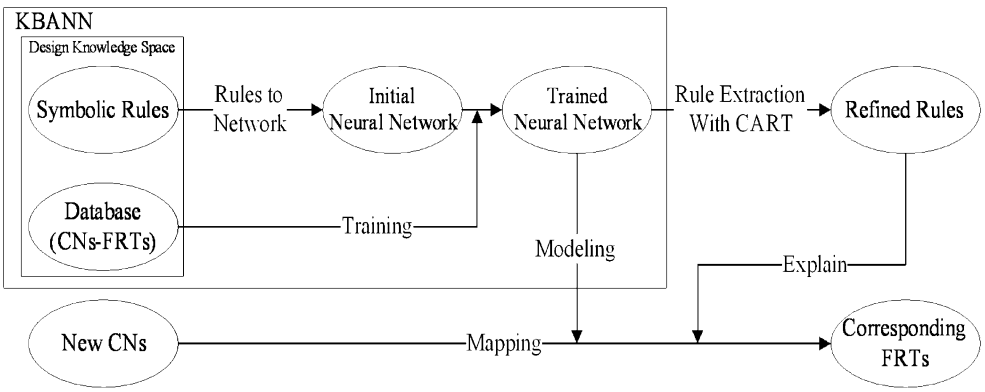


Fig. 2 The KBANN-DT method

the top of connectionist learning techniques [20]. It maps problem-specific "domain theories", represented in propositional logic, into a neural network (NN) and then refines this reformulated knowledge using backpropagation. The combination of both techniques performs better than either method alone, as the strengths of one method make up for the weaknesses of the other [30]. The superiority of KBANN is demonstrated in many fields, such as disease diagnosis [21], time series analysis [31], and customer relation management [32].

The process of KBANN consists of two steps. The first is a "rule translation" algorithm that maps the structure of an approximately correct domain theory into a neural network structure. The domain theory of product definition is derived from engineers. Originally they do not know the topology of the neural network. Following the rules-to-network algorithm of Towell and Shavlik [20], designers can construct the initial topology of a neural network. It prevents the network from learning from scratch. Then, the defined network is trained using the backpropagation learning algorithm. It refines the knowledge embedded in the network topology. The approach taken by KBANN is outlined in Table 2.

3.2 Construction of KBANN

3.2.1 Inputs, outputs, and initial neural network topology

In this paper, KBANN is used to model the mapping relationship between customer needs and functional requirement templates. Hence, the inputs of KBANN are customer needs and the outputs of KBANN are the coupled functional requirement templates, as well as the initial topology of KBANN as decided by domain theory.

Transform inputs It is well known that the performance of a neural network mainly depends upon the input features selected for its training [33]. In this research, the customer needs used as inputs are core needs from high-end customer segments [34]. To obtain what customers really want, Blecker proposed an advisory system to elicit customers' objective needs in mass customization [35]. Then, customer needs can be represented by a set of vectors $\{\mathbf{CV}_p\}$, $\mathbf{CV}_p=(cn_1, cn_2,..., cn_I)$.

The features of CN vectors, i.e., cn_i can be either numerical or nominal. In this research, numerical data is the data measured or identified on a numerical scale. An example is car/max speed (km/h) has values: 100, 150, and 200. Nominal data is the data that refers to named categories whose order is arbitrary. An example is car/color has values: white, red, blue and yellow.

Then, these CN features are transformed into the inputs for KBANN.

For a numerical CN feature, each feature is assigned to one input unit of KBANN. Its options are normalized into the range of [0, 1] following Eq. (1). In Eq. (1), cn_i^{ji} denotes the ith CN attribute that takes the j_ith option (each option corresponds to a numerical value), $cn_i^{ji}*$ denotes the normalized CN attribute cn_i^{ji}, and J_i denotes the option number of the ith CN attribute.

$$cn_i^{j_i}* = \frac{cn_i^{j_i} - \min\limits_{1\le j_i\le J_i}\left\{cn_i^{j_i}\right\}}{\max\limits_{1\le j_i\le J_i}\left\{cn_i^{j_i}\right\} - \min\limits_{1\le j_i\le J_i}\left\{cn_i^{j_i}\right\}}. \quad (1)$$

For a nominal CN feature, each of its options is assigned to one input unit of KBANN. The corresponding input value is set to 1, if one option is required by a customer. Otherwise, the input value is set to 0. For example, if CN feature car/color has three options: red, blue and black, then three input units: Color-is-red, color-is-blue, color-is-black, will be created. When a customer requires a red car, the values for the three input units can be: 1 0 0. As a particular case, if a nominal feature has only two options, both are assigned to only one input unit. The input value is set to 1, if one option is chosen. Otherwise, the input value is set to 0.

Outputs As the outputs of KBANN, each node corresponds to a product family, which can be denoted as a tuple: $\mathbf{FRT}k=(\boldsymbol{\mu}_k, \boldsymbol{\Delta}_k)$. Therefore, there is K output nodes as the design space of function domain is $\boldsymbol{\Omega}_{\mathbf{FRT}}=\{\mathbf{FRT}1, \mathbf{FRT}2,..., \mathbf{FRT}K\}$.

Initial neural network topology In this process, a symbolic knowledge encoding procedure is taken to translate domain theory (in the form of inference rules) into a network of threshold neurons using rules-to-network algorithm [20]. The procedure of constructing the initial topology of a KBANN is listed in Table 3. Towell's KBANN module only processes elementary production rules (IF...THEN rules) of order 0. Osorio and Amy extend it to accept the production rules of 0^+ [36]. The rule forms can be seen in Table 4.

Table 2 The learning procedure of KBANN

Framework	Description
Given	-A list of features to describe examples -An approximately correct domain theory with a set of rules -A set of examples
Procedure	-Map the domain theory structure into a neural network structure -Train the knowledge-based network with the classified examples -After training, the neural network can classify future examples

Table 3 The procedure of constructing the initial topology of a KBANN

Step	Description
1	Rewrite rules to eliminate disjuncts
2	Map rule structure into a neural network
3	Add important features not specified in mapping
4	Add hidden units to the neural network
5	Label units in the KBANN according to their level
6	Add links not specified by translation between all units in topologically-contiguous levels.
7	Perturb the network by adding near-zero random numbers to all link weights and biases.

Correspondingly, Fig. 3 illustrates the mapping method between rules and neural networks with some simple examples. In Fig. 3, B is the bias of a node and p can be any constant value. The initial weights of the links can be W or -W, where W is a positive value that is set empirically [20]. More precisely, Fig. 3a is an example for conjunctive rules. The link weight is set to W for a positive antecedent or -W for a negated antecedent. The bias on the consequent node (node corresponding to the rule's consequent) is set to (β-0.5) W, where β is the number of antecedents of an translated rule. Similarly, for disjunctive rules (see Fig. 3b), the link weights are set to W and the bias on the consequent node is set to 0.5W. As for the inequality (">" or "<"), the link weight is set to W or -W, whereas the bias of the consequent node is set to W*p or -W*p, respectively (see Fig. 3c and d). For complex rules, they can be decomposed into a set of simple rules and then mapped into a network [20].

Therefore, the resulting network is constructed with the rules. Before learning, the activation of the networks leads to the same results (outputs) as those obtained with the rules.

3.2.2 Neural network training

Once domain theory is translated into the initial topology of a neural network, KBANN is trained with historical transaction records. Using the backpropagation algorithm [37], KBANN refines its network via adjusting its link weights. To improve the performance in both training time and generalization, we use the cross-entropy error function-Eq. (2)-rather than the standard error function [20]. In Eq. (2), a_i is the activation of output unit i, d_i is the desired activation for unit i, and n is the number of output units.

$$Error = -\sum_{i=1}^{n} [(1 - d_i) * \log_2 (1 - a_i) + d_i * \log_2 (a_i)] \quad (2)$$

The training set for the neural network is $\mathbf{T} \sim \{(\mathbf{CV}_q, \mathbf{FRT}_q) \mid q=1, 2,..., Q\}$, which can be obtained from the company's sales records and product documentation. Through training, domain theory is revised and embodied in the links and weights of the neural network. Thus, the KBANN augments its network to learn concepts not provided by the initial rules.

3.3 Rule extraction from the trained KNANN network

After the construction of KBANN, the following step is to extract design knowledge from it. The extracted rules can help engineers comprehend the mapping relationship between CNs and FRs and scheme out suitable product specifications.

Here, the trained KBANN network is denoted as N°. The original data used for training KBANN is denoted as $\mathbf{T} \sim \{(\mathbf{CV}_q, \mathbf{FRT}_q) \mid q=1, 2,..., Q\}$. And the detailed procedures of rule extraction are illustrated as follows:

(1) New artificial data sets are generated. Initially, each feature vector $\mathbf{CV}_q$ in the original train set T is fed to the trained KBANN network N°, and then a class label $\mathbf{FRT}_q^{\alpha}$ is derived from the output of N°. Note that $\mathbf{FRT}_q^{\alpha}$ may not be identical to the original output, i.e., $\mathbf{FRT}_q$. By combining $\mathbf{CV}_q$ and $\mathbf{FRT}_q^{\alpha}$, a new data set $T^{\alpha} \sim \{(\mathbf{CV}_q, \mathbf{FRT}_q^{\alpha}) \mid q=1, 2,..., Q\}$ is generated.

(2) To enrich the resource for rule induction, more data is generated randomly. It is accomplished by randomly sampling the input (CN) space, and computing the target values for these sampled points by means of the trained KBANN network N°. For each randomly generated input vector $\mathbf{CV}_l^{\beta}$, if it is fed to N°, a functional requirement template $\mathbf{FRT}_l^{\beta}$ is the output from the network, $l \in \{1, 2,..., L\}$, where L is the number of the new data set that is generated. By combining $\mathbf{CV}_l^{\beta}$ and $\mathbf{FRT}_l^{\beta}$, the additional training

Table 4 Different rule forms

Rule type	Description	
Rules of order 0	Form	IF <Condition> (True/False) AND/OR <Condition> (True/False)...THEN <Conclusion>
	Example	IF B is False AND C is True THEN A
Rules of order 0^+	Form	IF <Feature> <Operator> <Value> AND/OR <Feature> <Operator> <Feature>...THEN <Conclusion>
	Example	IF B Greater Than 1.0 OR C Less Than 2.0 THEN A

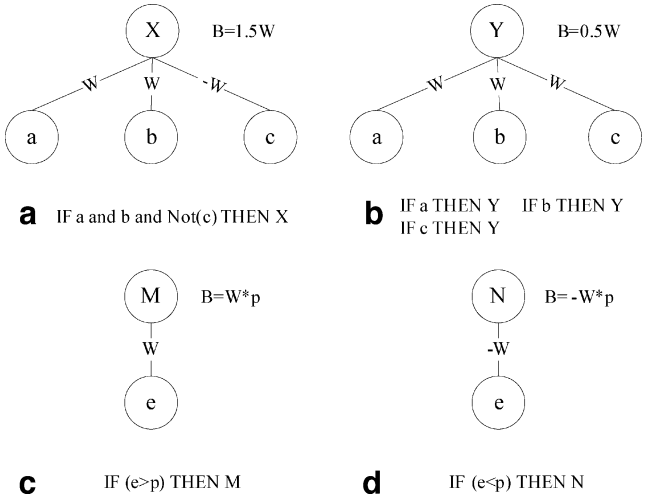


Fig. 3 The mapping method between rules and neural networks

data $T^{\beta}\sim\{(\mathbf{CV}_q^{\beta}, \mathbf{FRT}_l^{\beta}) \mid l=1, 2,..., L\}$ is generated. The size of T^{β}, L, is determined by experiments. Subsequently, both data sets are united as one set denoted by $T^{\gamma}=T^{\alpha}\cup T^{\beta}$, which is used as the train data for a decision tree algorithm.

As the KBANN algorithm usually has strong generalization ability [20], some bad ingredients of T, such as the noise, can be depressed by the process of KBANN. Therefore, the new data set T^{γ} is considered to be better than the original data set T for rule induction.

(3) Based on the new data set T^{γ}, a decision tree algorithm is used to extract rules. It can be obtained by means of rule induction algorithms such as ID3, C4.5, or CART [25]. In this research, classification and regression tree (CART) [24], a widely applied standard decision tree algorithm, is used for rule induction to understand the neural network. The splitting criterion chosen by the CART algorithm is Gini's diversity index [38]. The variant of CART used in this paper made use of minimal error complexity pruning with ten-fold cross-validation estimates.

4 Case study

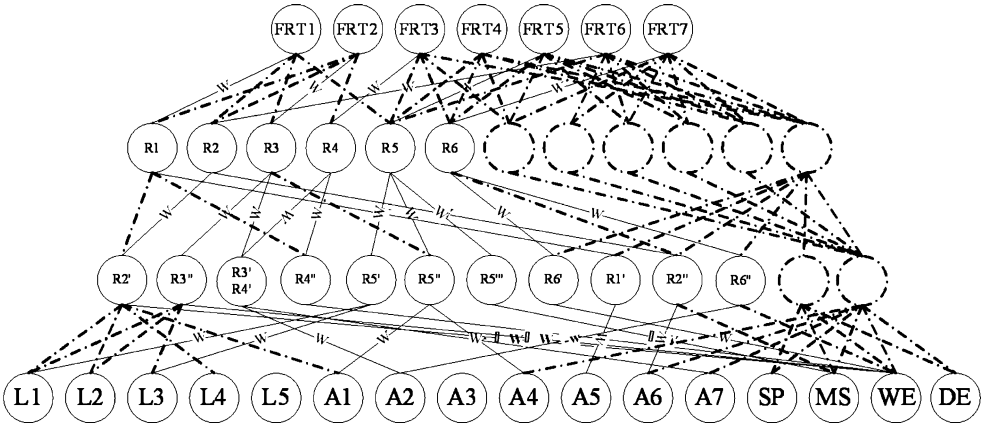
During the product definition phase in an elevator company, the KBANN-DT method is employed to facilitate the translation of a prioritized set of customer requirements into a set of functional requirements. Increasing demands for various elevators push manufacturers to increase product diversity while maximizing internal commonality. They develop new elevators by refining the existing elevators to meet different customer needs within a short time and low cost. In this research, the company had conducted extensive market researches and derived the data of customer expressions of various functionality related to elevators. These data have been collected from market surveys and analyzed based on natural language processing. Another resource of design knowledge, symbolic rules, is derived from design engineers with the method proposed in Sect. 3.2.1. As a result, two knowledge resources are utilized to modeling the relationship between customer needs and elevator specifications. Although this method cannot take the place of design engineers' creative work, it does help design engineers realize what customers really want or what

Table 5 Customer needs

CN feature			Option	NN	Code
cn_1	Description	Type			
cn_1	Location	Nominal	Hotel	L1	1,0
			Office	L2	1,0
			Mall	L3	1,0
			Hospital	L4	1,0
			Residence	L5	1,0
	Application	Nominal	Passenger	A1	1,0
			Cargo	A2	1,0
			Service	A3	1,0
			Tour	A4	1,0
			Medical service	A5	1,0
			Firefighters' service	A6	1,0
			Passenger-cargo	A7	1,0
cn_3	Speed m/s	Numerical	0.5,1.0,1.5,1.75,2.0,2.5,3.0	SP	0,0.2,0.4,0.5,0.6,0.8,1
cn_4	Floors	Numerical	2,3,...,40	MS	0,0.026,...1
cn_5	Car capacity(Kg)[a]	Numerical	320,400,...,5000	WE	0,0.017,...,1
cn_6	Decoration	Nominal	luxury,regular	DE	1,0

[a] Car capacity can also be represented by the number of passengers, e.g., 13 person=900 Kg, 15 person=1000 Kg.

Fig. 4 The topology of KBANN



product can keep high customer satisfaction with the least loss of commonality.

Investigation and analysis on the elevator market shows that the main concerns of CNs can be summarized into six categories as described in Table 5. In this table, $cn_i \mid \forall i \in [1, 2, ..., 6]$ denotes each CN feature for elevators. For example, cn_1 (i.e., "location") means where an elevator is installed, cn_4 (i.e., "floors") means the stops of an elevator, etc. Among these six features, designers can find that the 'location' 'application' and 'decoration' are of nominal type, while all the rest are numerical variables. The column "option" provides the options of each CN feature, the column "NN" refers to the KBANN's input nodes (see Fig. 4) corresponding to the options in the column "option", and the column "code" offers the numerical form of each option for KBANN. The numbers in the column "NN" and "code" are achieved following the transforming method mentioned in Sect. 3.2.1 (1).

Correspondingly, the company developed seven product families to meet different market segments. As listed in Table 6, each product family is characterized by nine specifications from the functional perspective. In this research, only the elevators with machine room are investigated. Indeed, the machine-room-less elevators can also be managed in a similar way.

Then, sales records containing customer needs and their related FR templates are taken as the train set. As illustrated in Table 7, there are 272 transaction records in the case study, i.e., $\mathbf{T} \sim \{(\mathbf{CV}_q, \mathbf{FRT}_q) \mid q = 1, 2, ..., 272\}$, where $\mathbf{CV}_q$ contains six customer attributes from Table 5 and $\mathbf{FRT}_q$ can be any one of the seven FR templates in Table 6. The first column of Table 7 is the customer serial number and the following columns are customer needs and related solutions (FR templates).

As another knowledge resource, initial symbolic rules that are derived from the experts are summarized in Table 8. Hence, domain theory is represented as a set of production rules. There are a total of six rules for KBANN. Each rule states the relationship between CNs and FR templates. Rule 1 means, for example, *if* the application of a

Table 6 FR templates and relative functional requirements

FR Template		Functional requirement								
		Load (kg)	Speed (m/s)	Max floor	Max height (m)	Door opening width (mm)	Car area (m^2)	Motor power (KW)	Door opening type	Motor control
FRT1	center value	1000	1.42	24	80	1100	3.45	13	Side or center	VVVF
	variation range	0	1~1.75	0	0	0	0	11~15	0	0
FRT2	center value	2071	0.57	10	30	As specified	As specified	15.14	Up-down	VWF
	variation range	1000~3000	0.5~1	0	0	0	0	11~18	0	0
FRT3	center value	3083	0.83	10	30	As specified	As specified	23.5	Up-down	VWF
	variation range	2000~5000	0.5~1	0	0	0	0	18~28	0	0
FRT4	center value	573	1.33	20.31	71	800	1.45	7.92	Center	VVVF
	variation range	320~700	1~1.75	16~24	60~80	0	0.9~1.75	4.5~11	0	0
FRT5	center value	900	1.25	21.71	74	871	2.35	10.54	Center	VVVF
	variation range	750~1350	1~1.75	16~24	60~80	800~1100	1.89~3.42	7.5~15	0	0
FRT6	center value	967	1.85	29.33	87	900	2.41	16.17	Center	VVVF
	variation range	900~1000	1.5~2.5	24~40	80~100	0	2.16~2.76	13~22	0	0
FRT7	center value	1483	1.67	31.47	84	1100	3.40	21	Center	VVVF
	variation range	135~1600	1~2.5	24~40	75~100	0	3~3.89	15~30	0	0

Table 7 Transaction records of CNs-FRT

C	Location	Application	Speed (m/s)	Floors	Car capacity (Kg)	Decoration	FRT
1	Mall	Medical service	1.5	10	1000	Regular	**FRT**1
2	Mall	Passenger	1.75	21	1350	Regular	**FRT**7
3	Office	Cargo	2.5	14	2000	Regular	**FRT**2
4	Mall	Cargo	2.5	36	2000	Regular	**FRT**2
5	Residence	Passenger	1	12	900	Regular	**FRT**5
6	Mall	Passenger	2.5	24	1350	Regular	**FRT**7
7	Hotel	Passenger	1.5	14	1000	Luxury	**FRT**5
8	Mall	Passenger	1	13	900	Luxury	**FRT**6
...	...	...	...	...	...	...	...
272	Residence	Passenger-cargo	1	14	4000	Regular	**FRT**3

customized elevator is 'medical service' *then* the corresponding elevator can be developed on the basis of **FRT**1. From another point of view, it means that the functional requirements in **FRT**1 can satisfy the market segment of 'medical service elevator'. Obviously, these rules are not enough for the designers as they can only identify less than half of the existing examples.

Consequently, the initial topology of KBANN is constructed following the seven steps listed in Table 3. Firstly, the six symbolic rules listed in Table 8 are directly mapped into the original structure of a network. Using the translation method described in Sect. 3.2.1 (3), a four-layer network is constructed (see the nodes and links with continuous lines in Fig. 4). It has 16 inputs and 7 outputs for the CN features and FR templates separately. Corresponding to the six rules, it has 11 nodes in the first hidden layer and 6 nodes in the second hidden layer. In this research, we set the continuous link weights equal to 1 (i.e., w=1). Then, according to the experiments, additional nodes and links (dash-and-dot ones in Fig. 4) are added to the network at expert-specified levels. It enables the network to learn knowledge not specified with the initial six rules. Based on our experiments, 2 nodes in the first hidden layer and 6 nodes in the second hidden layer are added to the network. Links are subsequently added to make the network fully connected between layers. The weight values of these links are equal to small random values.

Therefore, the resulting network is a four-layer full-connected neural network with topology: 16-13-12-7. The network in Fig. 4 is not a full-connected neural network as many of the dash-and-dot links are omitted for a better observation of the continuous links.

The developed KBANN is then trained by the adapted backpropagation procedure for learning and revision of rules until it identifies the training patterns at a high accuracy. As shown in Table 9, the transaction records from Table 7 is transformed into the inputs and outputs for KBANN.

To evaluate the performance of KBANN with a small number of training data, we compare the results by KBANN with those by two neural networks constructed without domain theory. One, denoted as Sta-ANN, is a standard three layer neural network. It has 16 inputs and 7 outputs separately corresponding to the CNs and **FRT**s of elevators. To specify the hidden nodes, we experiment with different ANN topologies: 16-x-7 | $\forall x \in$ [1, 2, 3, ..., 30] where x is the number of hidden nodes. Their performances are evaluated with the same training/test data from the elevator transaction records. The ANN with 16 hidden nodes performs as well as, or better than, all other ANNs with different hidden nodes. Then, the topology of Sta-ANN is set to: 16-16-7. The other, denoted as Str-ANN, has the same structure as KBANN: 16-13-12-7. However, its link weights and biases are not set using prior knowledge as KBANN. In fact, its link weights and biases randomly

Table 8 Initial symbolic rules

Rule	Rule content
1.	IF Application=Medical service THEN **FRT**1
2.	IF Application=Firefighters' service AND Car capacity<1300 THEN **FRT**6
3.	IF (Application=Cargo OR Application=Passenger-cargo) AND Car capacity<2500 THEN **FRT**2
4.	IF (Application=Cargo OR Application=Passenger-cargo) AND Car capacity>2500 THEN **FRT**3
5.	IF (Location=Hotel OR Location=Mall) AND (Application=Passenger OR Application=Tour) AND Floors<11 THEN **FRT**5
6.	IF Car capacity>1300 AND Application≠Medical service THEN **FRT**7

Table 9 Transformed records for NN

C	cn_1	cn_2	cn_3	cn_4	cn_5	cn_6	FRT
1	0 0 1 0 0	0 0 0 0 1 0 0	0.4	0.211	0.145	0	1 0 0 0 0 0 0
2	0 0 1 0 0	1 0 0 0 0 0 0	0.5	0.500	0.220	0	0 0 0 0 0 0 1
3	0 1 0 0 0	0 1 0 0 0 0 0	0.8	0.316	0.359	0	0 1 0 0 0 0 0
4	0 0 1 0 0	0 1 0 0 0 0 0	0.8	0.895	0.359	0	0 1 0 0 0 0 0
5	0 0 0 0 1	1 0 0 0 0 0 0	0.2	0.263	0.124	0	0 0 0 0 1 0 0
6	0 0 1 0 0	1 0 0 0 0 0 0	0.8	0.579	0.220	0	0 0 0 0 0 0 1
7	1 0 0 0 0	1 0 0 0 0 0 0	0.4	0.316	0.145	1	0 0 0 0 1 0 0
8	0 0 1 0 0	1 0 0 0 0 0 0	0.2	0.289	0.124	1	0 0 0 0 0 1 0
...	...	...	...	...	...	...	...
272	0 0 0 0 1	0 0 0 0 0 0 1	0.2	0.316	0.786	0	0 0 1 0 0 0 0

distribute around zero as a classical ANN. This network brings forward whether the structure of KBANN is responsible for its strength, i.e., if the structure of KBANN is better suited to learning in the elevator case than a standard neural network. Both networks are full connected neural networks trained using backpropagation algorithm [37]. Since learning from data alone, we expect the performance of each network to be inferior to that of KBANN, given the same number of training samples.

Then, the three neural networks are evaluated with the elevator data set, i.e., 272 samples listed in Table 9. In one trial, a certain number of samples, denoted as train-set-size, are randomly selected from the data set as the training samples. The remaining samples are used as testing samples. Each neural network is then trained and tested 10 times. Its average result is recorded as the final result. In this research, we run the trials over the three networks with train-set-size ranging from 10 to 250 (i.e., training/test data is 10/262, 30/242, 50/222... or 250/22). The average results are shown in Figs. 5 and 6.

Figure 5 shows the average error rate of the KBANN, Sta-ANN and Str-ANN, with train-set-size ranging from 10 to 250. The results are divided in three intervals as follows:

(1) *Train-set-size*>230 In this interval, the three networks have a similar performance. All of them can achieve a high accuracy because there are a lot of examples for learning.
(2) 80<*train-set-size*≤230 As train-set-size decreases, the performance gap between KBANN and the other two networks becomes obvious.
(3) *Train-set-size*≤80 All of the three networks could not perform well as the train-set-size is too small for learning.

Simultaneously, the corresponding standard deviation (STD) of the three networks with different train-set-size can be seen in Fig. 6. It is obvious that the STD of KBANN is the smallest across different train-set-size. From the comparison, we can see that the KBANN algorithm is more stable and robust than the other two networks.

The comparisons show that the KBANN network consistently outperforms the other two networks regardless of the train-set-size. KBANN can identify product definition patterns at a relative high accuracy, even if the training data becomes small (as the interval (2) of Fig. 5). It indicates that the symbolic rules embodied in KBANN improve its performance. Rather than from symbolic rules or transaction data only, the power of KBANN comes from both knowledge resources. Thus the KBANN network is able to learn design knowledge from the symbolic rules and historical data to improve the quality of the decisions made in the early stage of elevator design.

To have a good understanding of the decision making process, a decision tree algorithm is used to elicit knowledge from the fine tuned KBANN network. In this research, a standard decision tree algorithm, CART, is employed to generate binary decision trees. With the method proposed in Sect. 3.3, decision trees can be obtained from KBANN as illustrated in Fig. 7. Each leaf in the figure is labeled with a FR template and each interior node is labeled with a test build upon one attribute (CN). Likewise, each path in the tree can be easily translated into a symbolic rule. For

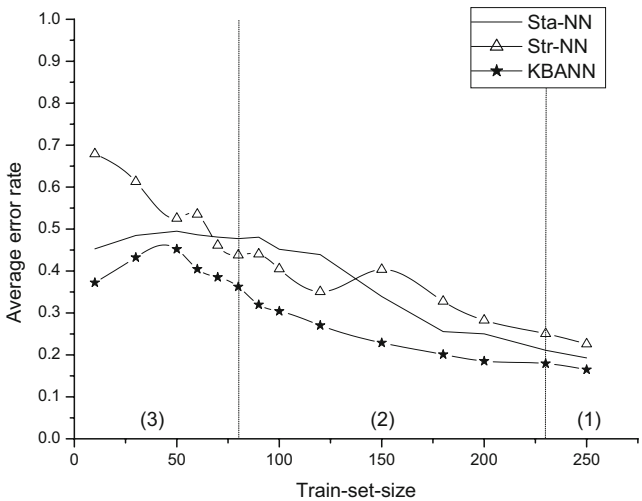


Fig. 5 Performance of KBANN, Sta-ANN and Str-ANN for different train-set-size

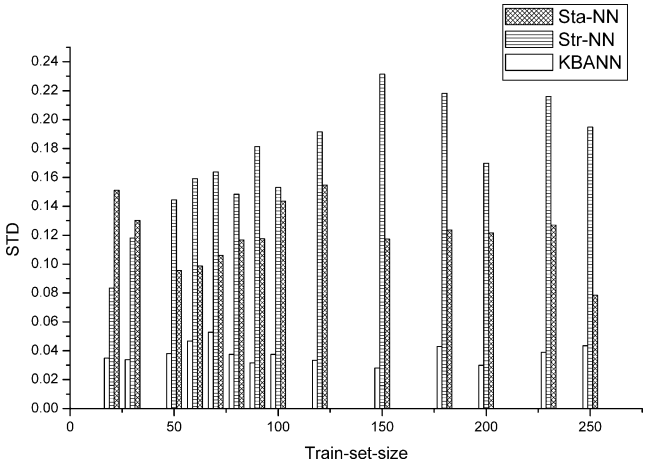


Fig. 6 STD of KBANN, Sta-ANN and Str-ANN for different train-set-size

example, a path from the root node to the node 8 (leaf) in Fig. 7 can be translated as: IF Car capacity<1275 AND Application≠Medical service AND Floors<10.5 THEN **FRT**5 is chosen. Therefore, KBANN-DT is able to illustrate internal logic between elevator families and customer needs.

The performance of KBANN-DT and CART-Ori is compared on 272 samples from the elevator company. Here, CART-Ori stands for developing a decision tree from the original 272 samples with the standard CART algorithm. Ten-fold cross validation [39] is performed on this case study. Firstly, 272 samples are divided into ten subsets with similar sizes (i.e., each has 27 or 28 samples). Secondly, each subset is used in turn for testing while the remainder is used for training. Thirdly, the whole procedure is repeated 10 times and its average result is taken as the final result. In out experiments, the number of additional training data T^{β} is the same as the original train set. After the experiment, the average error rate and standard deviation rate of KBANN-DT is 0.1707 and 0.0612, CART-Ori 0.2119 and 0.0666. The result indicates that the generalization ability of KBANN-DT is about 19% ((0.2119-0.1707)/0.2119=0.194) better than that of CART-Ori on the elevator case. It also indicates that the stabilization of KBANN-DT is about 8% ((0.0666-0.0612)/0.0666=0.081) better than that of CART-Ori. These results show that the KBANN-DT method can capture better design knowledge from two knowledge resources, that is, the rules derived from KBANN-DT are more reliable and accurate.

Through previous discussion and comparison to the traditional machine leaning techniques (NN and CART), we have found that the KBANN-DT method provides an efficient and effective means to identify product definition patterns. With this method, a company is able to make proper decision at the very beginning of the design process. When new customer needs arise, the company can locate their corresponding elevator families with KBANN. Then, design engineers work out product specifications with the support of the extracted rules. For example, a new customer needs appears, such as (mall, passenger, 1.75, 22, 1600, regular), which can be transformed into an input vector (0 0 1 0 0 1 0 0 0 0 0 0 0.5 0.526 0.274 0). Through calculating in KBANN, the last output node, **FRT**7, is activated. It is a reasonable output as customer 2 and 6 (in Table 7), who have similar CNs, also aim at **FRT**7. Then, designers can derive product specifications from the center vector and variation range of **FRT**7. The decision tree (Fig. 7) is used to facilitate the design engineers to narrow down the consideration set and derive a feasible solution quickly.

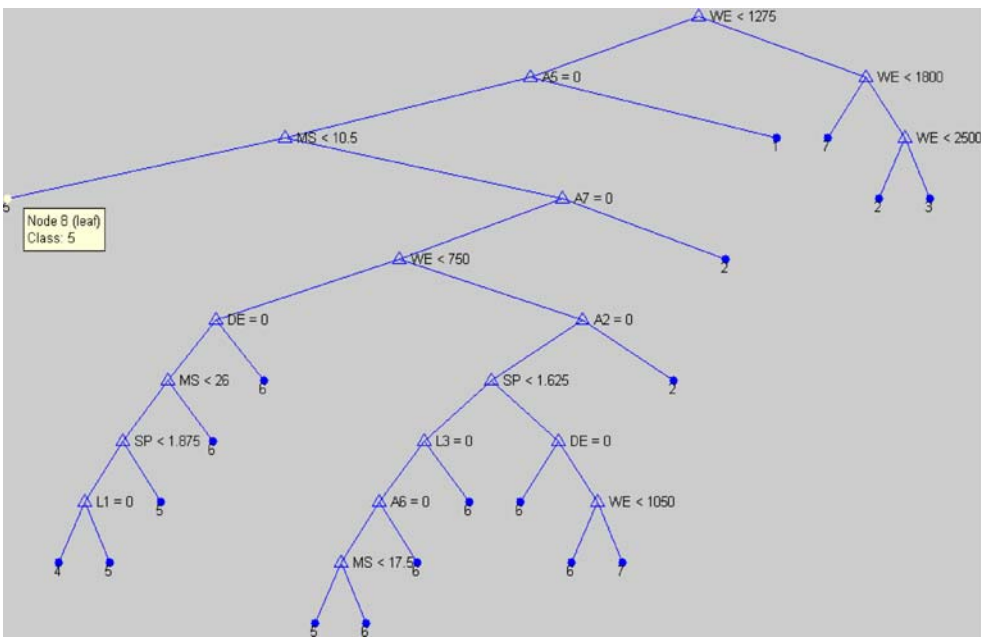


Fig. 7 An illustration of a decision tree

5 Conclusion

Catching the voice of customers and translating it into product specifications is a crucial issue for a successful product design. This paper proposed a learning-based hybrid method to facilitate the product definition process. In this method, a KBANN network is applied to modeling the relationship between customer needs and FR templates. The KBANN algorithm combines an inductive machine learning algorithm and deductive method to learn from examples and rules. The learning capability of KBANN improves the accuracy of the decision activities in PD. Based on the trained KBANN, a standard decision tree algorithm CART is used to extract rules from it. The extracted rules provide engineers with product design guidelines. To demonstrate the effectiveness of the proposed method, a case of an elevator design is conducted. The results show that the proposed method can be a promising tool in PD.

Future work is required to improve the proposed method in two aspects. The first is to efficiently utilize the extracted knowledge to achieve product specifications. The second is to combine manufacturing strategies and production technologies in the product definition stage.

Acknowledgement This research is supported by the National Natural Science Foundation of China / Hong Kong Research Grants Council (Grant No.70418013)/RGC Ref.: N-HKUST625/04; National Natural Science Foundation of China (Grant No.70471022, Grant No.70501021).
The authors would like to express their sincere thanks to the professor Mitchell M. Tseng, Dr. Wang Shijin for their valuable advices.

References

1. Pine BJ (1993) Mass customization: the new frontier in business competition. Harvard Business School Press, Boston MA
2. Tseng MM, Jiao J (1997) A variant approach to product definition by recognizing functional requirement patterns. Comput Ind Eng 33(3–4):629–633
3. Pugh S, Gardiner KM (1991) Total design: integrated methods for successful product engineering. Addison Wesley, Wokingham
4. Tseng MM, Jiao J (2001) Mass customization. In: Salvendy G (ed) Handbook of industrial engineering, 3rd edn. Wiley, New York, pp 684–709
5. Tseng MM, Jiao J (1998) Computer-aided requirement management for product definition: a methodology and implementation. Concurr Eng Res Appl 6(2):145–160
6. Du X, Jiao J (2001) Architecture of product family: fundamentals and methodology. Concurr Eng Res Appl 9(4):309–325
7. Tseng MM, Du X (1998) Design by customers for mass customization products. CIRP Ann - Manuf Technol 47(1):103–106
8. Du X, Jiao J, Tseng MM (2003) Identifying customer need patterns for customization and personalization. Integr Manuf Syst 14(5):387–396
9. Jiao J, Chen C-H (2006) Customer requirement management in product development: a review of research issues. Concurr Eng Res Appl 14(3):173–185
10. Wu H-H, Liao AYH, Wang P-C (2005) Using grey theory in quality function deployment to analyse dynamic customer requirements. Int J Adv Manuf Technol 25(11–12):1241–1247
11. Wu H-H, Shieh J-I (2006) Using a Markov chain model in quality function deployment to analyse customer requirements. Int J Adv Manuf Technol 30(1–2):141–146
12. Chen C-H, Khoo LP, Yan W (2002) A strategy for acquiring customer requirement patterns using laddering technique and ART2 neural network. Adv Eng Inf 16(3):229–240
13. Du X, Jiao J, Tseng MM (2006) Understanding customer satisfaction in product customization. Int J Adv Manuf Technol 31(3–4):396–406
14. Nagamachi M (2002) Kansei engineering in consumer product design. Ergon Des 10(2):5–9
15. Yan W, Chen C-H, Shieh M-D (2006) Product concept generation and selection using sorting technique and fuzzy c-means algorithm. Comput Ind Eng 50(3):273–285
16. Chen Z, Wang L (2006) Product definition in mass customization adopting neural network. Accepted by The 32nd Annual Conference of the IEEE Industrial Electronics Society, Paris, FRANCE, 7-10 November
17. Le Riche R, Gualandris D, Thomas JJ, Hemez F (2001) Neural identification of non-linear dynamic structures. Sound Vibr 248 (2):247–265
18. Jiao J, Zhang Y (2005) Product portfolio identification based on association rule mining. Comput Aided Des 37(2):149–172
19. Shao X-Y, Wang Z-H, Li P-G, Feng C-XJ (2006) Integrating data mining and rough set for customer group-based discovery of product configuration rules. Int J Prod Res 44(14):2789–2811
20. Towell GG, Shavlik JW (1994) Knowledge-based artificial neural networks. Artif Intell 70(1–2):119–165
21. Sordo M, Buxton H, Watson D (2001) A hybrid approach to breast cancer diagnosis. ftp://acl.icnet.uk/pub/PUBLICATIONS/sordo/chapter2001.pdf
22. Li C, Xu J, Xue L (2001) Knowledge-based artificial neural network models for finline. Int J Infrared Millim Waves 22(2):351–359
23. Towell GG, Shavlik JW (1993) Extracting refined rules from knowledge-based neural network. Mach Learn 13(1):71–101
24. Breiman L (1993) Classification and regression trees. Chapman & Hall, Boca Raton
25. Schmitz GPJ, Aldrich C, Gouws FS (1999) ANN-DT: an algorithm for extraction of decision trees from artificial neural networks. IEEE Trans Neural Netw 10(6):1392–1401
26. Boz O (2002) Extracting decision trees from trained neural networks. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 456–461
27. Andrews R, Diederich J, Tickle AB (1995) Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowl-Based Syst 8(6):373–384
28. Yao JT (2005) Knowledge extracted from trained neural networks - What's next? In: Proceedings of SPIE - The International Society for Optical Engineering, vol 5812, Data mining, intrusion detection, information assurance, and data networks security 2005, pp 151–157
29. Krishnan R, Sivakumar G, Bhattacharya P (1999) Extracting decision trees from trained neural networks. Pattern Recogn 32 (12):1999–2009
30. Towell GG (1991) Symbolic knowledge and neural networks: insertion, refinement and extraction. Ph.D. thesis, University of Wisconsin, Madison
31. van Zyl J, Omlin CW (2001) Knowledge-based neural networks for modelling time series. In: Proc 6th International Work-Conference on artificial and natural neural networks: bio-inspired applications of connectionism, pp 579–586
32. Haddawy P, Ha V, Restificar A, Geisler B (2004) Preference elicitation via theory refinement. J Mach Learn Res 4(3):317–337

33. Srivastava L, Singh SN, Sharma J (1999) Knowledge-based neural networks for voltage contingency selection and ranking. IEE Proc. Gen Transm Distrib 146(6):649–656
34. Krishnan V, Gupta S (2001) Appropriateness and impact of platform-based product development. Manag Sci 47(1):52–68
35. Blecker T, Abdelkafi N, Kreutler G, Friedrich G (2004) An advisory system for customers' objective needs elicitation in mass customization. In: The 4th International ICSC Symposium on Engineering of Intelligent Systems, University of Madeira, Funchal, Portugal
36. Osorio FS, Amy B (1999) INSS: a hybrid system for constructive machine learning. Neurocomputing 28:191–205
37. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by error propagation. Parallel Distributed Processing: Explorations in the Microstructure of Cognition 1:318–362
38. Sen PK (2005) Gini diversity index, hamming distance, and curse of dimensionality. Metron - Int J Stat LXIII(3):329–349
39. Zhou Z-H, Jiang Y (2003) Medical diagnosis with C4.5 Rule preceded by artificial neural network ensemble. IEEE Trans Inf Technol Biomed 7(1):37–42