https://doi.org/10.20965/jaciii.2006.p0682

Paper:

An Application of Binary Decision Trees to Pattern Recognition

Noboru Takagi

Department of Intelligent Systems Design Engineering, Toyama Prefectural University
5180 Kurokawa, Imizu, Toyama 939-0398, Japan
E-mail: noboru.takagi@ieee.org

[Received January 11, 2006; accepted February 20, 2006]

Decision rules are a key technique in decision making, data mining and knowledge discovery in databases. We introduce an application of decision rules, handwriting pattern classification. When decision rules are applied to pattern recognition, one rule forms a hyperrectangle in feature space, i.e., each decision rule corresponds to one hyperrectangle. This means that a set of decision rules is considered a classification system, called the subclass method. We apply decision rules to handwritten Japanese character recognition, showing experimental results.

Keywords: decision rule, off-line handwriting recognition, subclass method, pattern recognition

1. Introduction

Decision rules and decision trees are key techniques in data mining and knowledge discovery in databases [1–4]. A major advantage of decision rules and decision trees is their understandability and interpretability. Once we find a set of decision rules for a given data, so rules enable us to easily understand what obtained rules mean. Rough set theory provides a good way for finding decision rules from a given data-set, and it is often applied practically to KANSEI engineering.

Kudo et al. [5, 6] introduced the subclass method to pattern recognition because (1) it is assumed that objects with the same class must be distributed closely in feature space, and (2) if we can determine regions that cover all target objects, but cover none of no-target objects, then such a set of regions becomes a classification system. Based on this idea, Kudo introduced several algorithms to find regions that cover all target objects. In these algorithms, each region is found as a hyperrectangle of feature space. Hyperrectangles are an expression of different from decision rules or decision trees, i.e., a strong relationship exists between data mining with decision rules and trees and pattern recognition using the subclass method. This also implies that rough set theory is closely connected to pattern recognition.

Little discussion has been made on the practical application of subclass methods, so we first apply the subclass method to a practical application, off-line handwritten Japanese character (kanji) recognition, on which considerable research has been done [8–10]. Researchers have discussed classification systems using statistical techniques such as the subspace method, the simple similarity method, the multiple similarity method, the compound Mahalanobis function, and quadratic compound function. We start by discussing on handwritten kanji recognition based on logical techniques, which have the advantages of (1) results that make it what the obtained classification means, and (2) knowledge enabling us to know regions in which learning samples are distributed. A hyperrectangle is found by using the common rough set theory techniques such as LERS and RSES [11, 12]. This is difficult to apply practically to pattern recognition, it cannot handle a data-set with a large number of samples. To find decision rules such as hyperrectangles, we focus on decision trees learning because it requires less computation time when we use it in practical applications.

This paper is organized as follows: Section 2 discusses the subclass method, and the relationship between it and decision rules. Section 3 gives the algorithm for finding decision rules from a large learning sample set. This algorithm is based on binary decision tree learning. Section 4 goes into handwritten kanji classification. Section 5 gives experimental results of classification, and Section 6 presents conclusions.

2. Subclass Method and Decision Rules

The subclass method was first introduce by Kudo et al. [5] in 1989. The subclass method is motivated as follows:

Consider a class and its learning sample set $D^+ = \{x_1, x_2, \ldots, x_p\}$, and consider another class and its learning sample set $D^- = \{y_1, y_2, \ldots, y_q\}$. Here, it is assumed that D^+ and D^- are disjoint to each other, i.e., $D^+ \cap D^- = \emptyset$. Any element of D^+ is called a positive sample and of D^- a negative sample. If a region of feature space covers only positive samples, the region is called a subclass of positive samples. The basic idea of the subclass method is that if we can find regions in feature space that satisfy two properties

(1) regions cover all positive samples, and

(2) they do not cover any negative samples

then the set of regions is a classification system. Regions

express the area in feature space in which all positive samples are distributed. Specifically, regions should satisfy the following two properties:

(1) Exclusiveness: All positive samples must belong to at least one of the subclasses, and none of the negative samples may belong to any subclass for positive samples and

(2) Maximization: Each subclass must include as many positive samples as possible.

It is vital to the subclass method that how we can find subclasses that satisfy exclusiveness and maximization. Kudo el al. introduced an algorithm to find subclasses in which each subclass is expressed by a hyperrectangle in feature space, formulated as follows:

Let feature space be d-dimensional Euclidean space R^d, and let X be a subset of R^d. Then, Rect(X) is defined as the axis-parallel hyperrectangle spanned by X:

$$\mathrm{Rect}(X) = \left[\min_{x\in X} x_1, \max_{x\in X} x_1\right] \times \cdots \times \left[\max_{x\in X} x_n, \max_{x\in X} x_n\right],$$

where $x = (x_1, x_2, \ldots, x_d)$ and $x \in X$. A problem of the subclass method based on hyperrectangle is to find a subset family of positive sample set D^+ that satisfies exclusiveness and maximization.

Figure 1 shows the subclass method based on hyperrectangles. The hyperrectangle of the 3 subclasses found are

$$\mathrm{Rect}(X_1) = [t_1, t_3] \times [s_3, s_5],$$
$$\mathrm{Rect}(X_2) = [t_5, t_6] \times [s_2, s_5], \text{ and}$$
$$\mathrm{Rect}(X_3) = [t_2, t_4] \times [s_1, s_4].$$

These hyperrectangles are expressed by the following decision rules:

$\mathrm{Rect}(X_1)$:
If $t_1 \le x_1 \le t_3$ and $s_3 \le x_2 \le s_5$, then Class is Positive.
$\mathrm{Rect}(X_2)$:
If $t_5 \le x_1 \le t_6$ and $s_2 \le x_2 \le s_5$, then Class is Positive.
$\mathrm{Rect}(X_3)$:
If $t_2 \le x_1 \le t_4$ and $s_1 \le x_2 \le s_4$, then Class is Positive.

The subclass method based on hyperrectangles is closely related to the generation and reduction of decision rules.

Kudo et al. did not apply the subclass method to practical problems, because finding subclasses with both exclusiveness and maximization is difficult. We introduce binary decision trees to the subclass method because binary decision tree learning has the following advantages:

(1) Learning of a binary decision tree is fast even if the learning sample set is large.

(2) A binary decision tree expresses decision rules with many different classes, i.e., a binary decision tree need not to be calculated for every class.

(3) Our binary decision tree learning algorithm is based on the divide-and-conquer algorithm, and applied to parallel and distributed computing.

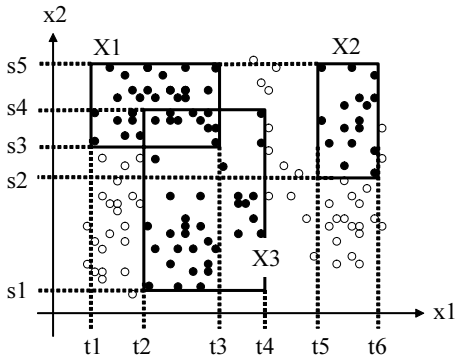


Fig. 1. Example of subclass method: Black circles denote positive samples and white circles negative samples. Rectangles denote subclasses for positive samples.

3. Binary Decision Trees and Subclasses

How, then do we find subclasses as hyperrectangles in feature space? The paper selects binary decision trees to find subclasses. The structure of a binary decision tree is either

(1) a *leaf*, including a class, or

(2) a *decision node* that specifies some test to be carried out on a single attribute value, with one branch and one subtree for each possible outcome of the test.

A binary decision tree is used to classify a case by starting at the root and moving down through it until a leaf is encountered. At each no-leaf decision node, the case's outcome for the test at the node is determined and attention shifts to the root of the subtree corresponding to this outcome.

Note that decision trees generally have the property that, for every path from the root to a leaf, an attribute appears at most only once, meaning no attribute appears twice on a path. Our aim is to find subclasses from a learning sample set efficiently, so in our binary decision trees, an attribute may appear more than once on a path from the root to a leaf.

The algorithm for generating a binary decision tree is as follows:

Algorithm: Create Binary Decision Tree (CBDT)

Input: Learning sample set D, with n attributes, m objects, and c classes.

Output: A binary decision tree that classifies learning sample set D.

Step 1: If learning sample set D is occupied by only one class i, then create a leaf node and attach class i to the leaf. Then return the leaf node.

Step 2: For any attribute a_i, calculate its classification contribution $GR(a_i, D)$. Set $a_{\max}$ as the attribute whose classification contribution is maximum among all n attributes.

Step 3: Divide learning sample set D into two parts D^L and D^R based on the calculation of $GR(a_{\max}, D)$.

Step 4: Apply this algorithm to D^L recursively, then create subtree T^L corresponding to D^L. Also apply this algorithm to D^U recursively, then create subtree T^R corresponding to D^R.

Step 5: Create a decision node and attach attribute $a_{\max}$ to the decision node. Also attach two subtrees T^L and T^R to the left and right sons of the decision node.

The measurement of classification contribution is calculated by Shannon's information. Let D be a learning sample set, D_i be a learning sample set of class i, and $p_i = |D_i|/|D|$, then the quantity of information of D, denoted by $I(D)$, is calculated by the following formula:

$$I(D) = -\sum_{k=1}^{c} p_i \log_c p_i \qquad \ldots\ldots\ldots \quad (1)$$

where $0 \leq p_i \leq 1$ for any class i, and c is the number of classes.

Based on formula (1), the classification contribution in CBDT is calculated as follows: Let D be a learning sample set with n attributes $\mathscr{A} = \{a_1, \ldots, a_n\}$, m objects $\mathscr{O} = \{o_1, \ldots, o_m\}$, and c classes $\mathscr{C} = \{1, \ldots, c\}$.

(1) Let a_i be an attribute of $\mathscr{A}$.

(2) For each $j = 1, \ldots, m$, which is the number of objects, do the following steps:

(2.1) First, divide D into two parts

$$D^L_{i,j} = \{o \in \mathscr{O} \mid o(a_i) \leq o_j(a_i)\}, \text{ and}$$
$$D^R_{i,j} = \{o \in \mathscr{O} \mid o_j(a_i) < o(a_i)\},$$

where o_j is the j-th object and $o(a_i)$ is the attribute value of a_i for object o. It is evident that $D^L_{i,j} \cap D^R_{i,j} = \emptyset$ and $D^L_{i,j} \cup D^R_{i,j} = D$.

(2.2) Calculate $I(D)$, $I(D^L_{i,j})$ and $I(D^R_{i,j})$.

(2.3) Calculate

$$GR_j(a_i, D) = I(D) - \left(\frac{|D^L_{i,j}|}{|D|} \cdot I(D^L_{i,j}) + \frac{|D^R_{i,j}|}{|D|} \cdot I(D^R_{i,j}) \right).$$

(3) $GR(a_i, D)$ is defined as the maximum value of $GR_j(a_i, D)$'s, i.e.,

$$GR(a_i, D) = \max\{GR_1(a_i, D), \ldots, GR_m(a_i, T)\}.$$

In Step 3 of CBDT, subsets D^L and D^R are subsets $D^L_{i,j}$ and $D^R_{i,j}$ whose $GR(a_i, D)$ is maximum among $GR_1(a_1, D), \ldots, GR_m(a_i, D)$.

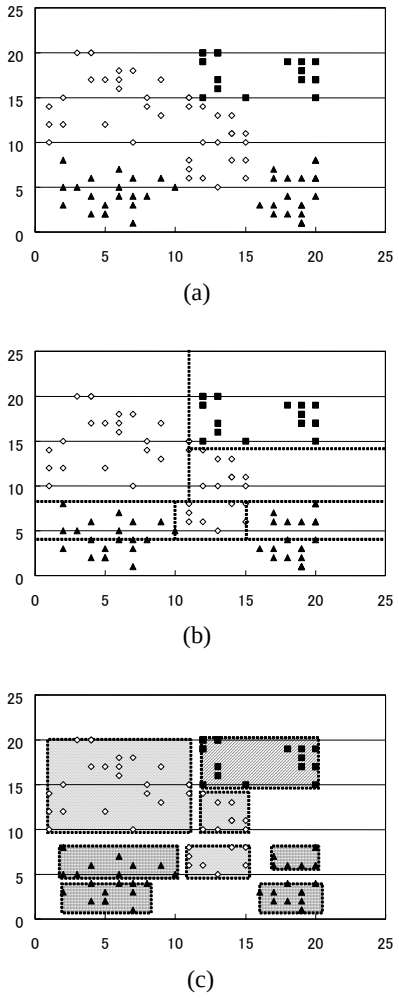


Fig. 2. Learning sample set in example 1: (a) is the learning sample set; class 1, 2, and 3 samples are denoted by white circles, squares, and triangles. (b) shows the 7 regions corresponding to the 7 decision rules obtained by the CBDT algorithm. (c) shows regions corresponding to reduced decision rules.

4. Handwritten Kanji Classification System

This section describes the process of our handwritten kanji classification. First, consider the following simple example of generating binary decision trees.

Example 1: Consider the learning sample set in **Fig.2(a)**. Learning samples are distributed on 2-dimensional plane, and the learning sample set contains three classes. Each class is denoted by a white circle, a square, and a triangle. Algorithm CBDT finds 7 decision rules for this learning sample set. The region corresponding to a decision rule is shown in **Fig.2(b)**. **Fig.3** is the binary decision tree obtained by CBDT.

A decision rule in a binary decision tree may include a large area that is not filled with samples, so we reduce the size of each decision rule by finding the maximum and minimum boundaries of each attribute as follows:

Assume we have the decision rule

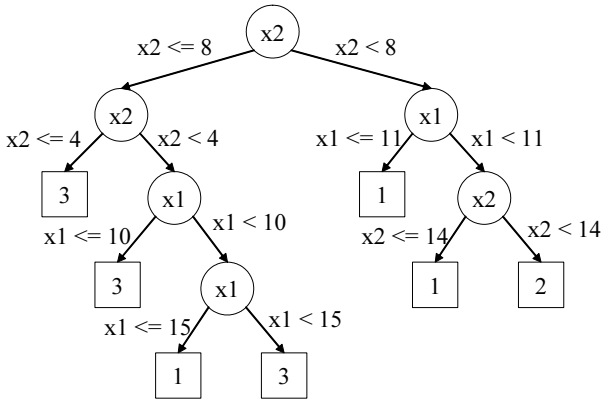


Fig. 3. Binary decision tree in example 1.

If $t_1^L \leq x_1 \leq t_1^R$ and ... and $t_n^L \leq x_n \leq t_n^R$, then class is i.

Let D' be the subset of learning sample set D such that elements of D' are covered by the decision rule. The decision rule is reduced by referencing subset D':

If $s_1^L \leq x_1 \leq s_1^R$ and ... and $s_n^L \leq x_n \leq s_n^R$, then class is i,

where $s_k^L = \min_{(x_1,\ldots,x_n)\in D'} x_k$ and $s_k^R = \max_{(x_1,\ldots,x_n)\in D'} x_k$. **Fig.2(c)** shows regions corresponding to the reduced decision rule in Example 1.

Our classification has five steps – scanning, preprocessing, feature extraction, classification generation, and recognition.

Scanning: A handwritten kanji character is scanned as a binary image.

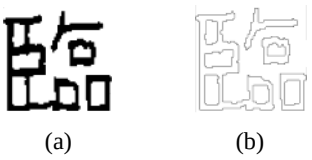
Preprocessing: The binary image is normalized, and the normalized image is outlined. Normalizing here means character size normalization.

Feature Extraction: The image obtained from preprocessing is translated into an improved directional element feature [8], which is a feature vector with 196 attributes.

Classification Generation: A classification is generated as a binary decision tree, then each decision rule is reduced by referencing the learning samples covered by the decision rule.

Recognition: An unknown kanji image is classified as class i when the distance between the feature vector of the kanji image and a decision rule of class i is shortest.

In preprocessing, the size of an original binary image is normalized into a 256×256 pixel binary image by enlarging or reducing it linearly. For the normalized image, we extract the outline of the handwritten character in the image (**Fig.4**).



(a) (b)

Fig. 4. (a) Normalized image and (b) outlined image.

Fig. 5. Three examples of ETL8.

Table 1. ETL8 specifications.

Number of Kanji Characters	881 characters
Number of Samples	160 par character
Data Format	64×63 pixel binary image

5. Experimental Results

Experimental results of handwriting recognition are discussed below.

We chose the ETL8 character database [13] as the benchmark data for evaluating our classification. The ETL8 character database was collected by the National Institute of Advanced Industrial Science and Technology, Japan. ETL8 is commonly used as benchmark data to compare the performance of off-line Japanese character recognition algorithms. ETL8 includes binary images of 881 Japanese kanji characters. Images are composed of 64×63 pixels. ETL8 has 160 sample images for each kanji character. **Fig.5** shows three sample images of ETL8, and **Table 1** lists ETL8 specifications.

To determine the classification rate, we consider the *leave-one-out* method [7], which conducts estimation by learning the entire data set except for the last sample and testing the last sample.

The following three sample sets are created from the ETL8 character database:

Data100: We select 100 kanji characters randomly, then sample set DATA100 becomes the set of all samples for the 100 kanji characters. Note that this sample set includes $100(\text{kanji characters}) \times 160(\text{samples}) = 16{,}000$ samples.

Data320: We select 320 kanji characters randomly, then sample set DATA320 becomes the set of all samples for the 320 kanji characters. Note that this sample set includes $320(\text{kanji characters}) \times 160(\text{samples}) = 51{,}200$ samples.

DATA881: This sample set includes all samples of ETL8, that is, it is identical to the ETL8 character database, so this sample set includes 140,960 samples.

Table 2. Experimental results.

DATA	A	B	C(%)	D(%)	E(%)	F(%)
100	16,000	1,014	93.7	3.86	100	78.0
320	51,200	5,851	88.6	5.67	96.9	70.6
881	140,960	26,557	81.2	7.37	94.1	55.8

"A" is the total number of evaluated samples, which equals to the product of the number of kanji characters and 160 (the number of testing sample sets). "B" is the total number of misclassified samples. "C" expresses A/B, the average classification of the 160 trials. "D" is the standard deviation of the 160 trials. "E" and "F" mean the best and worst classification rates among the 160 trials.

For sample set DATA100, 160 pairs of learning and testing sample sets are made as follows:

1. Select one sample randomly for each kanji character. Then, create a testing sample set with 100 samples (one sample for each kanji character). Let TEST1 be the testing sample set. The set of all remaining samples is the learning sample set corresponding to TEST1. Set $i \leftarrow 1$.

2. Set $i \leftarrow i+1$. Select one sample randomly for each kanji character so that selection satisfies the condition $\mathrm{TEST}i \cap \left(\bigcup_{j=1}^{i-1} \mathrm{TEST}j\right) = \emptyset$, i.e., no common sample exists between TESTi and all samples that selected before. The set of all remaining samples is the learning sample set corresponding to TESTi.

3. Repeat the above until i equals to 160.

For sample sets DATA320 and DATA881, we also created 160 pairs of learning and testing sample sets in a way similar to DATA100. Note that a leaning sample set includes 15,900 samples when it is created by DATA100, 50,880 by DATA320, and 140,079 by DATA881.

We conducted experiments for the learning and testing sample sets. Experiments involved 160 trials for each sample set DATA100, DATA320, and DATA881. Here, a trial means one pair of a learning process and a testing process. Results are summarized in **Table 2** and **Fig.6**.

Table 2 clarifies the following facts:

1. For any sample data set, there is a trial with more than 94% classification. When the number of kanji character is 100, we have a trial with 100% classification.

2. A trial with DATA881 has 55.8% classification, which is very low.

3. When the number of kanji character increases, the average classification rate decreases and standard deviation increases.

Based on the first of the above considerations, we conclude that our method have a relatively high classification rate for handwritten kanji character pattern recognition.

We considered why some samples were misclassified.

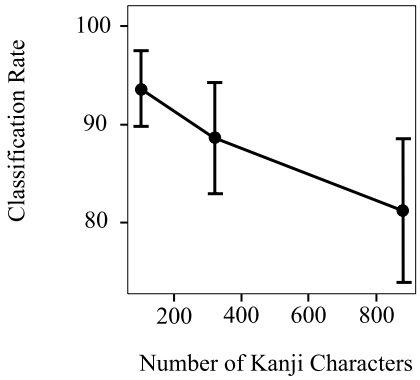


Fig. 6. Experimental results: average and standard deviation.

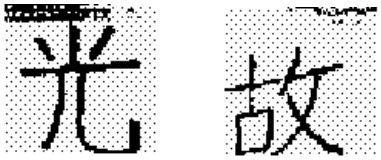


Fig. 7. Examples of images with noise.

(1) As shown in **Fig.7**, some sample images in ETL8 include a large number of noise pixels. Our method has no process that eliminates such noise pixels cleanly.

(2) In our method, an outline image is translated into a directional element feature vector. As shown in **Fig.4(b)**, outlining does not make an outline image whose line is smooth. This may present directional element feature vectors are not distributed compactly in feature space, even if their kanji characters are the same.

(3) A directional element feature vector contains 196 attributes, and each attribute take an integer from 0 to 800. This implies that the number of samples in a learning sample set is not enough to determine the ideal subclasses for every kanji characters.

6. Conclusions

We have discussed the application of decision rules to handwriting pattern recognition, focusing on handwritten Japanese kanji character recognition. We developed classification based on learning binary decision trees. We applied this to the ETL8 character database. Average classification rates was 93.7% for 100 kanji character, 88.6% for 320, and 81.2% for 881.

CPU time is several minutes in an environment in which the platform was IBM/AT Compatible (Pentium 3GHz) and the number of samples were 16,000. The system requires more than one hour to complete a learning process when samples number is about 140,000. CPU

time thus depends on the size of the binary decision tree, and increases exponentially with the number of samples. Parallel and distributed computing is one possible technique for solving this problem because the CBDT algorithm is based on the divide-and-conquer algorithm, which enables us to implement parallel and distributed processing for our method.

References:

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. H. Stone, "Classification and Regression Trees," Chapman & Hall, 1984.

[2] Z. Pawlak, "Rough Sets –Theory and Applications–," Kluwer Academic Publishers, Dordrecht, 1991.

[3] J. R. Quinlan, "C4.5 Programs for Machine Learning," Morgan Kaufmann Publishers, 1993.

[4] U. M. Fayyad, "Advances in Knowledge Discovery and Data Mining," MIT Press, 1996.

[5] M. Kudo and M. Shimbo, "Optimal Subclasses with Dichotomous Variables for Feature Selection and Discrimination," IEEE Trans. on Systems, Man, and Cybernetics, Vol.19, No.5, pp. 1194-1199, 1989.

[6] M. Kudo, Y. Torii, Y. Mori, and M. Shimbo, "Approximation of Class Regions by Quasi Convex Hulls," Pattern Recognition Letter, Vol.19, pp. 777-786, 1998.

[7] S. M. Weiss and C. A. Kulikowski, "Computer Systems That Learn – Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems," Morgan Kaufman Publishers, Inc., 1991.

[8] N. Sun, M. Abe, and Y. Nemoto, "A Handwritten Character Recognition System by Using Improved Directional Element Feature and Subspace Method," IEICE Trans. on Information and Systems, D-II, Vol.J78-D-II, No.6, pp. 922-930, 1995.

[9] T. Nakajima, T. Wakabayashi, F. Kimura, and Y. Miyake, "Accuracy Improvement by Compound Discriminant Functions for Resembling Character Recognition," IEICE Trans. on Information and Systems, D-II, Vol.J83-D-II, No.2, pp. 623-633, 2000.

[10] M. Suzuki, N. Kato, Y. Nemoto, and H. Ichimura, "A Discriminant Method of Similar Characters with Quadratic Compound Function," IEICE Trans. on Information and Systems, D-II, Vol.J84-D-II, No.8, pp. 1557-1565, 2001.

[11] J. W. Grzymala-Busse, "LERS – A System for Learning From Examples Based on Rough Sets," Handbook of Applications and Advances of the Rough Sets Theory, Kluwer Academic Publishers, pp. 3-18, 1992.

[12] J. G. Bazan and M. Szczuka, "RSES and RSESlib – A Collection of Tools for Rough Set Computations," Proc. of Rough Sets and Current Trends in Computing 2000, Lecture Note in Artificial Intelligence 2005, Springer-Verlag, pp. 106-113, 2001.

[13] http://www.is.aist.go.jp/etlcdb/

Name:
Noboru Takagi

Affiliation:
Department of Intelligent Systems Design Engineering, Toyama Prefectural University

Address:
5180 Kurokawa, Imizu, Toyama 939-0398, Japan

Brief Biographical History:

1991- Research Associate in Department of Electronics and Informatics, Toyama Prefectural University
1997- Associate Professor in Department of Electronics and Informatics, Toyama Prefectural University
2006- Associate Professor in Department of Intelligent Systems Design Engineering, Toyama Prefectural University

Main Works:

- N. Takagi, H. Kikuchi, and M. Mukaidono, "Applications of Fuzzy Logic Functions to Knowledge Discovery in Databases," Transactions on Rough Sets II –Rough Sets and Fuzzy Sets–, Lecture Notes in Computer Science, Vol.3135, Springer, 2004.

Membership in Learned Societies:

- The Institute of Electrical and Electronics Engineers (IEEE), Computer Society
- The Institute of Electronics, Information and Communication Engineers (IEICE)
- Information Processing Society of Japan (IPSJ)