Studies on Classifiers Based on Iteratively Composed Features

Kazuya HARAGUCHI

Studies on Classifiers Based on Iteratively Composed Features

Kazuya Haraguchi

Department of Applied Mathematics and Physics
Graduate School of Informatics
Kyoto University
Kyoto 606-8501, Japan

March, 2007

Doctoral Dissertation

submitted to Graduate School of Informatics, Kyoto University

in partial fulfillment of the requirement for the degree of

DOCTOR OF INFORMATICS

(Applied Mathematics and Physics)

Preface

Learning by machine. It has been a great dream of the human race, but has been a challenging issue at the same time. Among the mathematical learning problems studied so far, *classification* is one of the most fundamental but significant problems and has its application in many fields.

Classification is such a problem to find a function that is (approximately) equivalent with the hidden oracle function, by utilizing the given data set as a clue to the oracle. The problem was pointed out by Fisher from statistics in 1930's, and this research field was established by Rosenblatt's perceptron in 1960's.

A general approach to classification requires such a *representation model* that embodies a function by representing it as a logically structured object on computers, which is called a *classifier*. For this, there have been developed various representation models under their own assumption or concept on learning; e.g., linear discriminants, neural networks, decision trees, and support vector machines. A classifier is represented by mathematical expression or terminology. In some application, to extract new knowledge from the data set, it is important that its decision making process is understandable to people other than the resulting output (i.e., the decision made by a classifier).

In this thesis, we propose a new representation model called *iteratively composed features* (ICF). ICF models construction of classifiers as a process of iterative composition of meta-attributes, which we call *features*, based on subsets of the attributes (or dimensions, items, variables) that describe the considered data space. Starting with classification on binary data sets, we propose a construction algorithm of ICF classifiers through detailed analyses on features. We then extend the algorithm to more general data sets described with numerical and/or categorical attributes. The proposed algorithm is experimentally shown to have better performance than other previous algorithms on different representation models.

Not only does ICF provide a good classifier but also does it describe a hierarchy of concepts since it illustrates a decision process on how an abstract concept is formed by other concrete concepts or phenomena observed in the real world. The hierarchy of concepts is a different presentation of knowledge from typical decision trees. Hence, ICF suggests possibility of new methodologies in such fields as artificial intelligence, data mining, decision support systems, knowledge discovery, machine learning, and so forth. The author hopes that the research in this thesis will be helpful to advance the study in these significant and interesting fields.

March, 2007
Kazuya Haraguchi

Acknowledgement

This thesis would not have been possible without the help of many people, whom I would like to acknowledge here.

First of all, I am heartily grateful to Professor Hiroshi Nagamochi of Kyoto University for his enthusiastic guidance, discussion and persistent encouragement. He commented in detail on the whole work in the manuscript, which significantly improved the accuracy of the arguments and the quality of the expression. Without his considerable help, none of this work could have been completed.

I am deeply grateful to Professor Toshihide Ibaraki of Kwansei Gakuin University. His heartfelt and earnest guidance has encouraged me all the time. Several enthusiastic discussions with him were quite exciting and invaluable experience for me.

I am indebted to Professor Mutsunori Yagiura of Nagoya University, Professor Endre Boros of Rutger University, and Professor Hirotaka Ono of Kyushu University for numbers of helpful comments and suggestions.

I wish to express my gratitude to Professor Kazuhisa Makino of University of Tokyo, Professor Koji Nonobe of Hosei University, Professor Liang Zhao of Kyoto University, Professor Shunji Umetani of University of Electro-Communications, Professor Shinji Imahori of University of Tokyo, and all members in Professor Nagamochi's laboratory for many enlightening discussions on the area of this work and their warm friendship in the period I studied at Kyoto University.

I am also thankful to Professor Masao Fukushima of Kyoto University and Professor Yoshito Ohta of Kyoto University for serving on my dissertation committee.

Finally, but not least, I would like to express my heartiest gratitude to my family for their heartfelt cooperation and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Learning from examples

We consider the learning problem of acquiring knowledge on the underlying structure of a phenomenon as some representation form from a given set of its examples, samples or observations. A general approach to this problem consists of the following steps.

(L–1) Collection of examples. We choose the attributes, dimensions or items of the phenomenon to be investigated. We should select essential ones carefully in order to obtain meaningful knowledge after the learning step (L–3).

(L–2) Preprocess. We transform the examples into such a format (e.g., matrix) that is allowed as an input of the learning methodology specified in the next step (L–3).

(L–3) Learning. We select a learning methodology (e.g., classification problem, clustering, association rules, statistical analysis) and execute it on the preprocessed examples so as to acquire knowledge. The methodology should be selected by taking the application into account.

(L–4) Evaluation. The acquired knowledge is evaluated by some criteria. The evaluation may give feedback to the previous steps.

In this thesis, we mainly consider *classification problem*, which is a methodology in step (L–3). Classification problem is a typical learning issue and has its application in many fields such as statistics, artificial intelligence, machine learning, expert systems, cognitive science, pattern recognition, logical analysis of data (LAD), data mining, bio-informatics, and so on. To this significant learning problem, we propose a new approach based on *iteratively composed features* (ICF).

Note that the above steps (L–1) and (L–2) may have a great influence on the quality of the acquired knowledge regardless of the learning methodology. For example, as to (L–1), the amount of examples may be a crucial factor in improving the acquired knowledge; we would (resp., would not) achieve good knowledge with a large (resp., small) amount of examples. We also discuss this matter in this thesis.

1.1.1 Classification problem

Now we describe a mathematical framework of classification problem. In classification problem, we are given a *data space* $\mathbb{S}$, a set $\mathbb{C}$ of *classes*, an *oracle* y, and a *data set* Ω. For example, the data space $\mathbb{S}$ may be taken as the N-dimensional real space $\mathbb{R}^N$, the N-dimensional hypercube $\mathbb{B}^N = \{0,1\}^N$, or the set of direct products of N different sets. A member $\omega \in \mathbb{S}$ is called a *data element*. Each class in $\mathbb{C}$ is used to evaluate the quality or tendency of a data element ω. The oracle y is a function $y : \mathbb{S} \to \mathbb{C}$. The exact form of the oracle is not presented to us, but for some data elements in $\mathbb{S}$, their classes are available. An *example* ω is a data element such that the value $y(\omega) \in \mathbb{C}$ is available. The data set Ω is the set of such examples. Then classification problem asks us to find a function $c : \mathbb{S} \to \mathbb{C}$ that is an (approximately) equivalent function to y by utilizing the data set. In other words, we are required to find a function that predicts the class of a future example with a high accuracy.

Note that there are two types of classification problem in the literature: *supervised classification* and *unsupervised classification* [7, 81]. The problem described above corresponds to the supervised classification, in the sense that the oracle gives the classes of some data elements as a supervisor. On the other hand, the oracle is not assumed in the unsupervised classification, and the problem asks to partition the data set into subsets of "similar" examples. (The definition of similarity may differ in applications; e.g., Euclidean distance between examples in the real space.) The unsupervised classification problem is sometimes referred to as *clustering*.

In this thesis, we pay our attention to the *two-class supervised classification problem*, which forms a basis to a multi-class version of the problem. By taking $\mathbb{C} = \mathbb{B}$, we decompose the data set Ω into:

$$\Omega = \Omega^1 \cup \Omega^0,$$

where $\Omega^1 = \{\omega \in \Omega \mid y(\omega) = 1\}$ and $\Omega^0 = \{\omega \in \Omega \mid y(\omega) = 0\}$. We call an example $\omega \in \Omega^1$ (resp., $\omega \in \Omega^0$) a *true* example (resp., a *false* example). We call the error rate of a function

c on the data set Ω the *empirical error rate* of c, which is computed as follows;

$$e(c,\Omega) = \frac{1}{|\Omega|}(|\{\omega \in \Omega^1 \mid c(\omega) = 0\}| + |\{\omega \in \Omega^0 \mid c(\omega) = 1\}|). \tag{1.1}$$

If $e(c,\Omega) = 0$, then we say that c is *consistent* with Ω. We call the expected error rate of a function c on future examples the *true error rate*. (The true error rate is often referred to as *generalization error* in the literature.) Our purpose in the classification problem is to find such c that minimizes the true error rate.

We exploit the conventional approach to classification problem, where a function c is embodied by representing it as a logically structured object, called a *classifier*, which is easily implemented on computers. We say that a classifier *overfits* (resp., *underfit*) to the data set if the empirical error rate is too small (resp., large) and the true error rate is large (resp., still large). Overfitting (resp., underfitting) often occurs if the size of a classifier is too large (resp., small). The approach consists of three steps as follows.

(C–1) Selection of a *representation model*. A representation model gives the direction to represent c as a classifier. It consists of a *formulation framework* and a *classifying procedure*: The former gives how to formulate a logical structure of a classifier by mathematical terminologies. Then the latter is a procedure by which a classifier classifies an input data element based on its structure.

For example, let us take a representation model with hyperplanes, where we assume the data space to be $\mathbb{S} = \mathbb{R}^N$. In this case, a classifier is formulated by $N+1$ coefficients $w_0, w_1, \ldots, w_N$. A classifying procedure may assign the class 1 or 0 to an input data element $\omega \in \mathbb{S}$ according to whether $\sum_{q=1}^{N} w_q\omega_q - w_0 \geq 0$ or not.

(C–2) Design of a *construction algorithm*. For the selected representation model, a *construction algorithm* is designed. A construction algorithm outputs a classifier by utilizing information of the data set. The aim of this step is to design construction algorithms that can produce classifiers attaining small true error rates.

For example, a construction algorithm to hyperplane classifiers is to determine $N+1$ coefficients based on the data set. *Fisher's linear discriminant* and *perceptron* are well-known classical construction algorithms of hyperplane classifier [67, 69, 81, 84].

(C–3) Construction of a classifier. We then construct a classifier by applying the designed construction algorithm to the data set. The constructed classifier should be evaluated by true error rate. However, we do not usually know the oracle and thus cannot compute the true error rate exactly. Hence a classifier is evaluated by an estimate of the true error rate, which is obtained by a *resampling method* such as hold-out, cross validation, and bootstrapping [29, 81].

Among the above steps, selection of a representation model (C–1) is most important. Under an inappropriate representation model to the considered data space and oracle, any classifier does not achieve a small true error rate. For example, let us assume the data space to be $\mathbb{S} = \mathbb{R}^2$, the data set to be $X \subseteq \mathbb{S}$, and the oracle y to be a function as follows:

$$y(\omega) = \begin{cases} 1 & \text{if } \omega_1\omega_2 \leq 0, \\ 0 & \text{otherwise,} \end{cases}$$

i.e., y is an XOR (exclusive-or) function. One can observe that any hyperplane classifier does not attain a small true error rate even if it is consistent with the data set X [56, 81]. Thus hyperplane is not appropriate to this case. We should select the representation model carefully so that a good classifier will be delivered. Prior knowledge or experts of the considered data may help this task [33].

Also, it is important that the meaning of its decision making process (i.e., how a data element is classified by the classifying procedure) is understandable to people in some application area (e.g., automated diagnosis system in medical situation) [33]. In other words, the decision making process should be interpretable by means of natural language, propositional logic or pictorial representation. In some representation models, the decision making process is too mathematical for us to understand, and we do not gain any information from a classifier other than the output class. *Neural networks* (NN) or *support vector machines* (SVM) are examples of such representation models. For these models, a number of theoretical studies have been made since 1980's (including *back-propagation* [70], a construction algorithm for NN) [26, 45, 52], and their high performance (in terms of true error rate) in practical situations has also been observed [54, 59, 62]. However, due to the complicated decision making process, they should be applied to situations where the decision making process is not focused; e.g., pattern recognition.

On the other hand, *decision trees* (DT) is a representation model such that the meaning of the decision making process is rather easy to understand. Developed in 1970's, DT has been applied to such fields as artificial intelligence, data mining, expert systems, knowledge discovery, and so on [33]. In this thesis, we propose a new understandable representation model, called iteratively composed features (ICF). We will describe the overview of DT and ICF in the next Section 1.2.

1.1.2 Data size in learning

Intuitively, if a given data set is sufficiently large (compared with the entire data space), we expect that it contains enough information that enables us to acquire meaningful knowledge. For classification problem, if the data set is large, our task may be reduced to minimize the empirical error rate. Indeed, it is known that, for any representation model, if the data set has examples more than the threshold which is determined by the *VC dimension* (standing for Vapnik-Chervonenkis dimension) of the model, then the difference between the empirical error rate and the true error rate of a classifier becomes sufficiently small with a high probability [14, 22, 78].

VC dimension was proposed by Vapnik and Chervonenkis [80] at the end of 1960's[1] and has formed one of the central topics in the field of statistical learning theory [78]. However, it must be difficult to compute the VC dimension exactly for many practical representation models. From viewpoint of computational complexity, Papadimitriou and Yannakakis [63] introduced the problem of computing the VC dimension of a given model as the first LOGNP-complete problem (where LOGNP-complete is a weaker class of NP-complete). Besides, there are several studies to point out the complexity of this problem [57, 71, 74]. There are some attempts to derive upper or lower bounds on the VC dimension for some representation models; e.g., Bartlett [11], Baum et al. [12] and Maass [53] investigated such bounds for NN, where the details are summarized in the monograph by Anthony [6].

Researchers in such fields as *PAC* (*Probably Approximately Correct*) *learning* consider the classification problem under a particular condition (e.g., the class $y(\omega)$ of an example ω is flipped with a certain probability [5]) or even other learning problems (e.g., recognition of rectangles in the two dimensional plane [9]). They try to derive a necessary and sufficient size of the data set for the "success" of learning of their own definition [4, 10, 77]. Note that some of their analyses do not necessarily utilize VC dimension but is based on the fundamental probability theory. We will study the related problem later in Chapter 5.

1.2 Representation models for classification problem

Let us focus on the two-class classification problem on the data space $\mathbb{S} = \mathbb{B}^n$. In this section, we introduce two representation models ICF and DT.

[1]The original paper was written in Russian in 1968 [79].

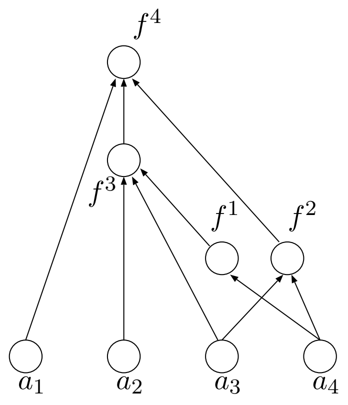


Figure 1.1: An ICF classifier

1.2.1 Iteratively composed features (ICF)

Here we introduce the representation model ICF which is our proposal in this thesis. Let us call a classifier by the representation model ICF an *ICF classifier*. An ICF classifier is based on a Boolean function which we call a *feature*. As a special case of feature, we introduce an *initial feature* $a_j : \mathbb{B}^n \to \mathbb{B}$ for each attribute $j = 1, 2, \ldots, n$, which is defined as $a_j(x) = x_j$ for $\forall x \in \mathbb{B}^n$.

Formulation framework.

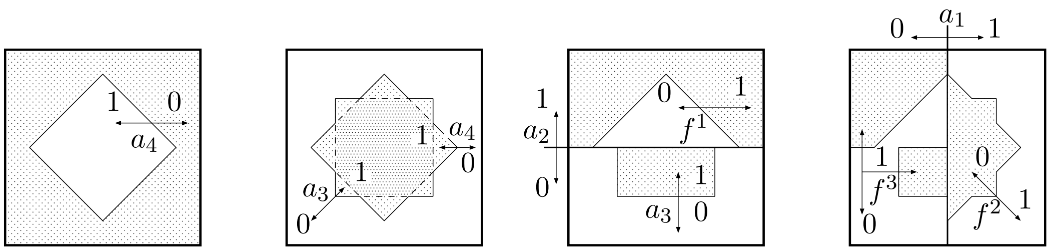
The structure of an ICF classifier with the *final feature* f is formulated by a directed acyclic graph (DAG), denoted by $G_f = (V_f, E_f)$, where V_f and E_f denote the sets of nodes and edges, respectively. Each node corresponds to one feature f_S, which is a Boolean function $f_S : \mathbb{B}^S \to \mathbb{B}$ of the set S of its fan-in features (i.e., S is the set of features connected to the node by the incoming edges). Among the nodes, there is only one node having no outgoing edges, which corresponds to final feature f. Also, the nodes having no incoming edges correspond to initial features.

In the rest of the thesis, we signify an ICF classifier by its final feature for convenience. Figure 1.1 shows an ICF classifier $f^4 = f_{\{a_1, f^3, f^2\}}$, where $f^3 = f_{\{a_2, a_3, f^1\}}$, $f^2 = f_{\{a_3, a_4\}}$ and $f^1 = f_{\{a_4\}}$. The height of an ICF classifier is defined as the length of the longest path from an initial feature to the final feature (e.g., in Figure 1.1, the height is 3).

For a set S of features and $x \in \mathbb{B}^n$, let us denote by $x|_S$ the projection of x onto S (e.g., if we take $S = \{a_3, a_4\}$, then $x^1|_S = (a_3(x^1), a_4(x^1)) = (1, 1)$ for the example x^1 in Table 1.1). For convenience, we write $f_S(x|_S)$ by $f_S(x)$. If $f_S(x) = y(x)$ holds, then we say that f_S *covers* x.

Table 1.1: A data set $X = X^1 \cup X^0$

		x_1	x_2	x_3	x_4
X^1	x^1	1	1	1	1
	x^2	0	1	1	0
X^0	x^3	1	0	1	0
	x^4	0	0	0	0
	x^5	0	0	0	0



$f^1 = \bar{a}_4$ $\quad$ $f^2 = a_3 \vee a_4$ $\quad$ $f^3 = \bar{a}_2 a_3 \vee a_2 f^1$ $\quad$ $f^4 = \bar{a}_1 f^3 \vee a_1 f^2$

Figure 1.2: Construction of features $f^1, \ldots, f^4$ in the data space $\mathbb{B}^4$

Classifying procedure.

An ICF classifier f classifies a data element $x \in \mathbb{B}^n$ into $f(x) \in \mathbb{B}$ by the following procedure:

Classifying Procedure of ICF

Input: An ICF classifier f with a DAG $G_f = (V_f, E_f)$ and a data element $x \in \mathbb{B}^n$.

Output: The class of x evaluated by f.

Step 1: For each initial feature $a_j \in A \cap V_f$, determine the output value $a_j(x)$ by $a_j(x) = x_j$. Let $V'_f := A \cap V_f$.

Step 2: Select a feature $f_S \in V_f \setminus V'_f$ such that $S \subseteq V'_f$ holds.

Step 3: Determine $f_S(x)$ based on the value of $g(x)$ for each $g \in S$.

Step 4: If f_S is the final feature, then output $f_S(x)$ and halt. Otherwise, let $V'_f := V'_f \cup \{f_S\}$ and return to Step 2.

In the above procedure, V'_f represents the set of features whose output values to x are determined. Let us show an example. Assume that $f^1, \ldots, f^4$ in Figure 1.1 are defined as such

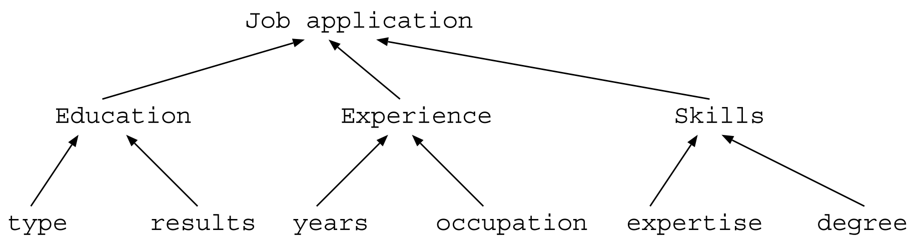


Figure 1.3: A concept hierarchy

Boolean functions as:

$$
\begin{aligned}
f^1 &= \bar{a}_4, \\
f^2 &= a_3 \vee a_4, \\
f^3 &= \bar{a}_2 a_3 \vee a_2 f^1, \\
f^4 &= \bar{a}_1 f^3 \vee a_1 f^2,
\end{aligned}
$$

respectively. Then the ICF classifier classifies the false example $x^3 = (1, 0, 1, 0)$ in Table 1.1 as follows: In Step 1, we determine $a_1(x^3) = 1$, $a_2(x^3) = 0$, $a_3(x^3) = 1$ and $a_4(x^3) = 0$ and have $V'_f = \{a_1, a_2, a_3, a_4\}$. In Step 2, we select $f^1 = f_{\{a_4\}}$ since $\{a_4\} \subseteq V'_f$ holds (i.e., $a_4(x^3)$ is already determined). Then in Step 3, we determine $f^1(x^3) = \bar{a}_4 = 1$. In Step 4, since $f^1 \neq f^4$, we update $V'_f = V'_f \cup \{f^1\}$ and return to Step 2. In this way, we determine $f^2(x^3) = 1$, $f^3(x^3) = 1$ and finally $f^4(x^3) = 1$. By checking all examples in this way, one can see $e(f^4, X) = 1/5$. Figure 1.2 illustrates how the final feature f^4 in Figure 1.1 is constructed in the data space $\mathbb{B}^n$ based on other features. Those subspaces labeled as 1 are depicted by shaded areas.

Proposition 1.1 *The computation time of Classifying Procedure of ICF is evaluated as* $O(|V_f| + |E_f|)$.

We may regard the decision making process of an ICF classifier as a phenomenon such that the conclusion of an abstract concept (or idea, knowledge) is determined based on other concrete concepts. Let us show Figure 1.3 as an example of such *concept hierarchy*. In the hierarchy, the conclusion of the final concept `Job application` is determined by three intermediate concepts: `Education`, `Experience` and `Skills`, which are also based on six attributes described at the bottom of the hierarchy.

Construction algorithm of an ICF classifier.

We wish to design such a construction algorithm that finds an ICF classifier attaining a small true error rate. Our strategy is to generate promising features iteratively based on the set A of initial features, and to obtain an ICF classifier by regarding one of them as the final feature. Let us define the operation of "composition" as follows.

Definition 1.1 *We say that a feature f_S is* composed *if* (1) *a set S of other features is selected and* (2) *the function $f_S : \mathbb{B}^S \to \mathbb{B}$ is determined.*

Then our strategy is summarized as follows. In the following, F denotes a set of the features available to composition, and is initialized as $F := A$.

<u>**Scheme for Constructing an ICF Classifier**</u>

Input: A data set X with n attributes (and thus $A = \{a_1, a_2, \ldots, a_n\}$).

Output: An ICF classifier f.

Step 1: Let $F := A$.

Step 2: Iterate the following until some specified conditions are satisfied: Compose a feature f_S for an arbitrary $S \subseteq F$, and let $F := F \cup \{f_S\}$.

Step 3: Output some $f \in F$ and halt.

In order to design a "good" ICF construction algorithm, we have to resolve several problems. For example:

- In the composition of a feature, how do we select a feature set S from F? As to the size of S, too large S may cause overfitting, and thus a reasonable sized S should be selected.
- For the selected S, how do we determine the Boolean function $f_S : \mathbb{B}^S \to \mathbb{B}$? There are $2^{2^{|S|}}$ possible functions as f_S.
- When should we stop iterating Step 2? An ICF classifier of too complex structure may overfit to the data set, and we may not need to examine ICF classifiers of too many nodes or too large heights.

Later in Chapter 3, we will discuss two construction algorithms of ICF classifiers based on the above scheme.

Related works with ICF.

Bohanec and Zupan [16] and Zupan et al. [85] proposed heuristic algorithms to construct a concept hierarchy from a discrete data set (i.e., an attribute can take only a discrete value). They aimed at constructing a hierarchy consistent with the data set, and have not taken true error rate into account. Also, their approach is based on recursive decomposition of the set A of initial features, while ours is on iterative composition of features.

An ICF classifier is also referred to as a *decomposable Boolean function* or a *modular function* in the field of LAD [19, 27]. In the literature, Boros et al. [17] considered the following problem.

Problem k-CONSISTENCY

Given: A data set $X = X^1 \cup X^0$ with n attributes and $k+1$ subsets $S_0, S_1, \ldots, S_k \subseteq A$ of initial features.

Question: Can we compose a feature $f = f_{S_0 \cup \{f_{S_1}, \ldots, f_{S_k}\}}$ consistent with X?

Note that S_κ and $S_{\kappa'}$ are not necessarily disjoint ($\kappa, \kappa' = 0, 1, \ldots, k$). They showed that k-CONSISTENCY is NP-complete for $k \geq 2$ and that one can solve the problem in $O(n|X|)$ time for $k = 1$.

Concentrating on the case of $k = 1$, Ono et al. [60, 61] considered the existence of a *nontrivial partition* (S_0, S_1) of the initial features (i.e., $S_0 \cup S_1 = A$, $S_0 \cap S_1 = \emptyset$, $|S_1| > 1$) such that $f_{S_0 \cup \{f_{S_1}\}}$ is consistent; i.e., the solution to 1-CONSISTENCY$(X, \{S_0, S_1\})$ is YES. The problem is summarized as follows.

Problem DECOMPOSABILITY

Given: A data set $X = X^1 \cup X^0$ with n attributes.

Question: Is there a nontrivial partition (S_0, S_1) of A such that the solution to 1-CONSISTENCY$(X, \{S_0, S_1\})$ is YES?

They showed that, unfortunately, the problem DECOMPOSABILITY is NP-complete. It means that it is NP-hard to find a nontrivial partition (S_0, S_1) that minimizes the empirical error rate $e(f_{S_0 \cup \{f_{S_1}\}}, X)$. These results suggest the difficulty of constructing an ICF classifier performing well on future examples.

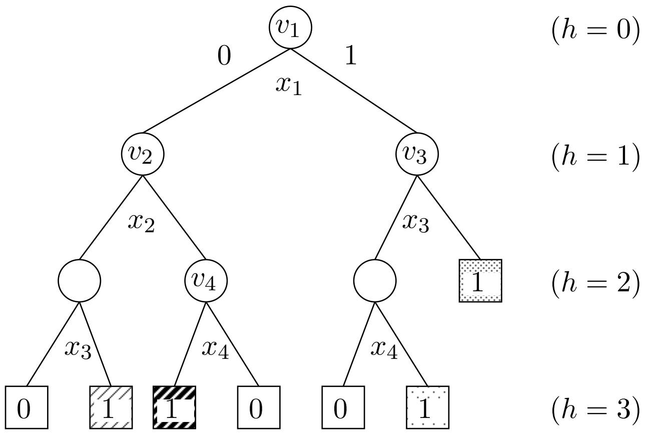


Figure 1.4: A decision tree

Bioch [13] studied the problem 1-CONSISTENCY when the input is given as $X = \mathbb{B}^n$ (i.e., we take the entire data space for X, and thus $|X| = 2^n$) and a nontrivial partition (S_0, S_1) of initial features. In this case, under a weak assumption on (S_0, S_1), it was shown that the problem becomes coNP-complete. While the above problems are defined on features, i.e., Boolean functions, Popova and Bioch [64] discussed the similar problems on general discrete data sets and discrete functions.

1.2.2 Decision trees (DT)

Formulation framework.

Let us denote by t a decision tree, a classifier $t : \mathbb{B}^n \to \mathbb{B}$. The structure of t is formulated by a rooted binary tree $G_t = (V_t, E_t)$ and a label $\ell_t : V_t \to \{0, 1, \ldots, n\}$, where V_t and E_t denote the sets of nodes and edges, respectively. We denote by V_t^{inner} and $V_t^{\text{leaf}} (= V_t \setminus V_t^{\text{inner}})$ the sets of inner nodes and leaves in V_t, respectively. Each inner node $v \in V_t^{\text{inner}}$ is associated with attribute $\ell_t(v) \in \{1, 2, \ldots, n\}$, and each leaf $v' \in V_t^{\text{leaf}}$ is labeled as $\ell_t(v') \in \{0, 1\} = \mathbb{B}$. The depth of a node v is defined as the number of edges in the path between v and the root of the tree. The height of G_t is defined as the length of the longest path from the root to a leaf and is denoted by $h(G_t)$. Let V_t^h denote the set of nodes with depth h, and let $V_t^{\text{inner},h}$ (resp., $V_t^{\text{leaf},h}$) denote the set of inner nodes (resp., leaves) with depth h. Figure 1.4 shows an example of decision trees.

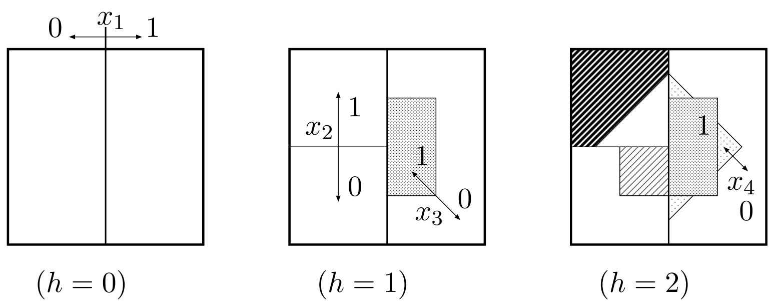


Figure 1.5: Construction of a decision tree in the data space $\mathbb{B}^4$

Classifying procedure.

For an inner node v of a rooted binary tree, let us denote by v_{right} (resp., v_{left}) the right (resp., left) child of v. A decision tree t classifies a data element $x \in \mathbb{B}^n$ into $t(x) \in \mathbb{B}$ as follows.

Classifying Procedure of DT

Input: A decision tree t with a graph $G_t = (V_t, E_t)$ and a label ℓ_t and a data element $x \in \mathbb{B}^n$.

Output: The class of x evaluated by t.

Step 1: Let v be the root of G_t.

Step 2: If v is a leaf, then output $\ell_t(v)$ and halt.

Step 3: Let $j := \ell_t(v)$. If $x_j = 1$, then let $v := v_{\text{right}}$. Otherwise, let $v := v_{\text{left}}$.

Step 4: Return to Step 2.

For example, the empirical error rate of the decision tree t in Figure 1.4 on the data set X in Table 1.1 is $e(t, X) = 1/5$.

Note that a node v in a decision tree corresponds to a subspace of $\mathbb{B}^n$. Let us denote by $\mathbb{B}^n(v)$ the subspace. In particular, the root v corresponds to the entire data space, i.e., $\mathbb{B}^n(v) = \mathbb{B}^n$. The children of an inner node v correspond to its disjoint subspaces, that is,

$$\mathbb{B}^n(v_{\text{left}}) = \{x \in \mathbb{B}^n(v) \mid x_j = 0\},$$
$$\mathbb{B}^n(v_{\text{right}}) = \{x \in \mathbb{B}^n(v) \mid x_j = 1\},$$

where $j = \ell_t(v) \in \{1, 2, \ldots, n\}$. Then the subspaces of leaves are labeled as either 0 or 1. For the decision tree in Figure 1.4, how the data space is divided and how the subspaces are labeled are illustrated in Figure 1.5, where those subspaces labeled as 1 are depicted by shaded areas.

Proposition 1.2 *The computation time of Classifying Procedure of DT is evaluated as* $O(|V_t|)$.

We mention how the decision making process of DT is interpreted. For DT, we see that a decision tree t classifies a data element $x \in \mathbb{B}^n$ according to *propositional rules* as follows: Assume that x visits the nodes $v^0, v^1, \ldots, v^h$ in the classifying procedure, where $v^{h'} \in V_t^{h'}$ ($h' = 0, 1, \ldots, h$), v^0 denotes the root, and v^h denotes a leaf. Let us take $b_0, b_1, \ldots, b_{h-1} \in \mathbb{B}$ as follows; if $v^{h'+1}$ is the left (resp., right) child of $v^{h'}$, then we set $b_{h'} = 0$ (resp., 1) for $h' = 0, 1, \ldots, h-1$. Then the decision making process is interpreted by a propositional rule such that: "If $x_{j_0} = b_0$, $x_{j_1} = b_1$, $\ldots$, and $x_{j_{h-1}} = b_{h-1}$ hold, then x should belong to the class $\ell_t(v^h)$," where $j_{h'} = \ell_t(v^{h'})$ for $h' = 0, 1, \ldots, h-1$.

Construction algorithm of a decision tree.

A typical algorithm for constructing a decision tree is described as follows.

Typical Algorithm for Constructing a Decision Tree

Input: A data set X with n attributes.

Output: A rooted binary tree $G_t = (V_t, E_t)$ and a label $\ell : V_t \to \{0, 1, \ldots, n\}$.

Step 1: Let $V_t := \{v\}$ and $E_t := \emptyset$. Determine $\ell_t(v) \in \mathbb{B}$.

Step 2 (Branching):

- **Step 2-1:** Choose a leaf v of the current tree $G_t = (V_t, E_t)$.
- **Step 2-2:** Add a pair of new nodes v_{left} and v_{right} to G_t as the children of v (i.e., $V_t := V_t \cup \{v_{\text{left}}, v_{\text{right}}\}$ and $E_t := E_t \cup \{(v, v_{\text{left}}), (v, v_{\text{right}})\}$). Determine the labels $\ell_t(v) \in \{1, 2, \ldots, n\}$ and $\ell_t(v_{\text{left}}), \ell_t(v_{\text{right}}) \in \mathbb{B}$. ($v$ now becomes an inner node.)
- **Step 2-3:** If $G_t = (V_t, E_t)$ does not satisfy specified conditions, then return to Step 2-1.

Step 3 (Pruning):

- **Step 3-1:** Choose two leaves $v_{\text{left}}, v_{\text{right}}$ having the same parent v and prune them; i.e., let $V_t := V_t \setminus \{v_{\text{left}}, v_{\text{right}}\}$, $E_t := E_t \setminus \{(v, v_{\text{left}}), (v, v_{\text{right}})\}$ and determine the label $\ell_t(v) \in \mathbb{B}$. (v now becomes a leaf.)

Step 3-2: If $G_t = (V_t, E_t)$ does not satisfy specified conditions, then return to Step 3-1.

Step 4: Output G_t and ℓ_t and halt.

The operation of branching in Step 2 enlarges the size of the decision tree by adding a pair of nodes as leaves of the current tree. On the other hand, the operation of pruning in Step 3 makes the decision tree smaller, and it is usually used to avoid the overfitting of the resulting decision tree.

Related works with DT.

For a given data set X, the problem of constructing a minimum height decision tree consistent with X is known to be NP-hard [39, 49]. A polynomial time $\ln |X|$-approximation algorithm was proposed by Arkin et al. [8], and Laber and Nogueira [50] showed that there is no $o(\log |X|)$-approximation algorithm unless P=NP.

As to decision tree construction, there have been proposed many construction algorithms such as C4.5 [65], SPRINT [73], and so on [36, 40, 82], where different methods of branching or pruning are used. Among these algorithms, we exploit C4.5 as a competitor of the proposed ICF classifier construction algorithms.

1.3 Overview of the thesis

In this thesis, we propose a new representation model ICF for the two-class supervised classification problem and establish a construction algorithm of ICF classifiers.

Before designing a concrete construction algorithm, we first examine whether ICF is a potential representation model or not in order to confirm that ICF can be adopted as a promising model. In (C–1) of Section 1.1.1, as observed in the example of hyperplane classifier and the XOR function, we need to select an appropriate representation model by taking the considered oracle (or even the underlying distribution of examples) into account. In other words, it is desirable for a representation model to realize a good classifier for many cases.

In Chapter 2, we discuss the potential performance of ICF as a representation model in comparison with DT. We conduct both theoretical and experimental analyses. For the theoretical analysis, we show the superiority of ICF to DT in view of *Occam's Razor* [15]. Occam's Razor asserts that not only should a classifier attain a small empirical error rate but also should it be represented in a compact form. Based on this assertion, we show that if DT has such a classifier for given data set X with an oracle y, then ICF also has such one, and that the converse does not hold. For the experimental analysis, we investigate whether a good ICF classifier (in the sense of true error rate) can be constructed for a given data set. In the composition of a feature f_S, as defined in Definition 1.1, we have to (1) select a subset $S \subseteq F$ and to (2) determine the Boolean function for f_S. We introduce two function determination schemes for the task (2), named *majorization* and *extended majorization*. We compare the entire sets of features composed by these schemes with that of decision trees, and show that the sets of features have a better classifier than that of decision trees in many cases.

Based on these results, following (C–2) of Section 1.1.1, we design construction algorithms of ICF classifiers. In Chapter 3, we propose two construction algorithms of ICF classifiers ALG-ICF and ALG-ICF*, which are based on majorization and extended majorization, respectively. We show that ALG-ICF* *can* construct a better classifier than C4.5 of DT or BSVM of SVM under properly tuned parameter values.

However, as is often the case with typical construction algorithms, determining appropriate parameter values is usually a difficult and time-consuming task. Furthermore, ALG-ICF* is a construction algorithm which works only on an $\mathbb{M}$-valued data set (i.e., $\mathbb{S} = \mathbb{M}^n$), where $\mathbb{M} = \mathbb{B} \cup \{*\}$ and $*$ denotes a *missing bit*. Hence the applicable situations of ALG-ICF* are rather limited.

To overcome these shortcomings, in Chapter 4, we extend ALG-ICF* to real world data sets consisting of numerical and/or categorical attributes. For this purpose, we incorporate a *discretization scheme* into ALG-ICF* as its preprocessor, by which an input real world data set is transformed into $\mathbb{M}$-valued one. We examine three discretization schemes, *domain based construction* (DC), *space based construction* (SC) and *integrated construction* (IC). Let us denote by ALG-ICF$^*_{\mathrm{DC}}$ (resp., ALG-ICF$^*_{\mathrm{SC}}$ and ALG-ICF$^*_{\mathrm{IC}}$) the construction algorithm ALG-ICF* equipped with DC (resp., SC and IC) as the preprocessor. We observe that ALG-ICF$^*_{\mathrm{IC}}$ constructs a better classifier than C4.5 on real world data sets without finely tuning parameter values. By this, we assert that ALG-ICF$^*_{\mathrm{IC}}$ is a new classifier construction algorithm displaying a concept hierarchy on real world data sets, and hope that it serves as an excellent knowledge discovery tool to practitioners.

In Chapter 5, we analyze the size of a data set used for learning. We treat a tractable structure of knowledge, called a *pattern*. For given data set $X = X^1 \cup X^0$ and constants $\theta_1, \theta_0 \in [0, 1]$, a (θ_1, θ_0)*-pattern* is defined as a combination of attribute values which appears in X^1 at least in proportion of θ_1 and in X^0 at most in proportion of θ_0. If θ_1 is "large enough" and θ_0 is "small enough," a (θ_1, θ_0)-pattern represents a trend of attribute values toward X^1 rather than X^0, and thus might be utilized as meaningful information of data.

One can enumerate all (θ_1, θ_0)-patterns in X in incrementally polynomial time [1]. If the data set is small, however, then some of the enumerated (θ_1, θ_0)-patterns may be deceptive; i.e., they do not reflect the underlying structure of data, and exist as (θ_1, θ_0)-patterns in X only by chance, due to its small size.

We observe necessary and sufficient sizes of a data set with which it contains such deceptive (θ_1, θ_0)-patterns with a low probability. A sufficient data size can be derived easily by utilizing classical technique of probability theory. On the other hand, the derivation of a good necessary size may be rather difficult. For this question, we propose a necessary data size by utilizing the idea arising from randomness: If the size is small, even a data set which is generated at random should contain (θ_1, θ_0)-patterns. Then we claim that a data set of such size should contain deceptive (θ_1, θ_0)-patterns. We justify our claim by computational experiments. We also derive an upper bound on the proposed necessary data size as its estimator.

Finally, in Chapter 6, we summarize our study in this thesis.

Chapter 2

Comparison of Potential Performance between ICF and Decision Trees

2.1 Introduction

In this chapter, we compare the potential performance between two representation models, ICF and DT. We conduct both theoretical and experimental analyses in Sections 2.2 and 2.3, respectively. We observe the superiority of ICF to DT through the analyses.

2.2 Theoretical analysis based on Occam's Razor

Assume that we are given a data set X labeled by the oracle Boolean function $y : \mathbb{B}^n \to \mathbb{B}$. The principle of Occam's Razor [15], serving as an important dialectics in many research fields, states that entities should not be multiplied beyond necessity. It suggests that the truth should be represented in a compact form. Following Occam's Razor, we make the next assumption on y.

Assumption 2.1 *We assume that the oracle y is a Boolean function that can be represented in a compact form by using terminologies of natural languages and/or mathematical expressions.*

E.g., if y is the parity function, then we may represent y by a brief English sentence as follows; "if a Boolean vector $x \in \mathbb{B}^n$ contains an odd number of 1, then $y(x)$ is 1, and otherwise, it is 0." Thus the parity function can be introduced as an oracle. On the other hand, if y is a function such that $y(x)$ is determined at random for all $x \in \mathbb{B}^n$, then there may be no way to represent y other than the naïve way to store $y(x)$ for all $x \in \mathbb{B}^n$. It requires the space complexity $O(2^n)$, which becomes exponentially large with the increase of n. We do not assume such a function to be the oracle.

Let us denote by R a representation model (which may indicate either ICF or DT) and by r a classifier of R (which may indicate either an ICF classifier f or a decision tree t). Based on this assumption on the oracle y, we consider that it is desirable for r not only to attain a small empirical error rate but also to be represented in a compact form; i.e., the space complexity needed to represent the structure of r is small. In order to treat this complexity, we introduce the *representation complexity* of r, denoted by $\gamma_R(r)$, which is defined as the length of the bit string needed to encode the structure of r by an *encoding scheme*. An encoding scheme is designed for each representation model R, and it encodes a classifier into a bit string. If $\gamma_R(r) = O(\text{poly}(n))$ holds (i.e., representation complexity is small), then we say that r is a *compact* classifier.

Let us say that r is an *ε-classifier* if $e(r, X) \leq \varepsilon$ holds for a constant $\varepsilon \in [0, 1]$. Then using these terminologies, we assert that r is desirable to be a compact ε-classifier for a small constant ε. This assertion is supported by the well-known principles of AIC (An Information Criterion) [2, 67] and MDL (Minimum Description Length) [41, 66, 68] from statistics, which state that, roughly speaking, a good classifier (in the sense of true error rate) is achieved by a moderate empirical error rate and representation complexity. In fact, it is intuitively understood that a classifier with a small (resp., large) empirical error rate and a large (resp., small) representation complexity may overfit (resp., underfit) to the data set X; both empirical error rate and representation complexity should be reasonably small.

The above assertion leads to the following statement: To be an appropriate representation model to a given tuple (ε, X, y), it is necessary that R has a compact ε-classifier. Thus R may serve as a general representation model if it can realize a compact ε-classifier for many tuples of (ε, X, y). From this point of view, we show that ICF is a superior representation model to DT, as summarized in the following theorem.

Theorem 2.1 *Assume that we are given a constant $\varepsilon \in [0, 1/2)$, a data set X labeled by an oracle y. If there exists a compact ε-decision tree, then there also exists a compact ε-feature. Note that the converse does not hold (i.e., even if there exists a compact ε-feature, there does not necessarily exist a compact ε-decision tree).*

The rest of this section is contributed to the proof of the above theorem. We define the representation complexity of ICF and DT by using the encoding schemes, which is described in Section 2.2.1. Then in Section 2.2.2, we give the entire proof of Theorem 2.1.

2.2.1 Representation complexity

ICF.

The representation complexity $\gamma_{\mathrm{ICF}}(f)$ of an ICF classifier f with $G_f = (V_f, E_f)$ is defined by using the following encoding scheme: Let $k_{\max}$ denote the maximum number of fan-ins allowed for all features in V_f. (E.g., we may take $k_{\max} = n$.) For each feature $f_S \in V_f \setminus A$, we need $\log_2 k_{\max}$ bits to identify the number of fan-ins $|S|$, and $2^{|S|}$ bits to identify the output values of $f_S(s)$, where $s \in \mathbb{B}^S$. Moreover, in order to identify the fan-ins S of f_S, a trivial way requires additional $|V_f|$ bits: if $g \in S$ for a feature $g \in V_f$, the corresponding bit is set to 1, and otherwise, it is set to 0. In this trivial way, $|S|$ out of $|V_f|$ bits are set to 1, and the other bits are set to 0. Assuming that we have a corresponding table of combinations, we consider that $\log_2 \binom{|V_f|}{|S|}$ bits are enough to identify the fan-in S of f_S. Finally, the complexity $\gamma_{\mathrm{ICF}}(f)$ for representing f is determined as follows:

$$\gamma_{\mathrm{ICF}}(f) = \sum_{f_S \in V_f \setminus A} \left(\log_2 k_{\max} + 2^{|S|} + \log_2 \binom{|V_f|}{|S|} \right). \tag{2.1}$$

DT.

The representation complexity $\gamma_{\mathrm{DT}}(t)$ for a decision tree t with $G_t = (V_t, E_t)$ and $\ell_t : V_t \to \{0, 1, \ldots, n\}$ is defined by using the following encoding scheme: For each node $v \in V_t$, 1 bit is required to identify whether v is an inner node or a leaf. Then if $v \in V_t^{\mathrm{inner}}$, additional $\log_2 n$ bits are required to identify the branching attribute $\ell_t(v) \in \{1, 2, \ldots, n\}$; otherwise (i.e., if $v \in V_t^{\mathrm{leaf}}$), additional 1 bit is required to identify the label $\ell_t(v) \in \mathbb{B}$. Since G_t is a binary rooted tree, $|E_t| = 2|V_t^{\mathrm{inner}}|$ and $|E_t| = |V_t| - 1$. Thus $|V_t^{\mathrm{leaf}}| = |V_t^{\mathrm{inner}}| + 1$ holds, and we have $|V_t^{\mathrm{leaf}}| = (|V_t| + 1)/2$ and $|V_t^{\mathrm{inner}}| = (|V_t| - 1)/2$. Finally, the representation complexity $\gamma_{\mathrm{DT}}(t)$ is determined as follows.

$$\gamma_{\mathrm{DT}}(t) = |V_t| + |V_t^{\mathrm{inner}}| \log_2 n + |V_t^{\mathrm{leaf}}| = \frac{3}{2}|V_t| + (\frac{|V_t| - 1}{2}) \log_2 n + \frac{1}{2}. \tag{2.2}$$

Before closing this subsection, we introduce some terminology on DT for the proof in the next subsection. Let X^v denote the set of examples in X that visit a node $v \in V_t$, and let $X^{v,1} = \{x \in X^v \mid y(x) = 1\}$ and $X^{v,0} = \{x \in X^v \mid y(x) = 0\}$. Consider two examples $x, x' \in X^v$ for an inner node v. We say that x and x' are *separated* at v if x and x' visit different children of v. Clearly, if a decision tree is a consistent classifier, any two examples $x^1 \in X^1$ and $x^0 \in X^0$ reach different leaves, and thus are separated at some inner node.

We assume that the index of an attribute $j \in \{1, 2, \ldots, n\}$ appears at most once in the path from the root to any leaf, since all data elements $x \in \mathbb{B}^n$ encountering the second test

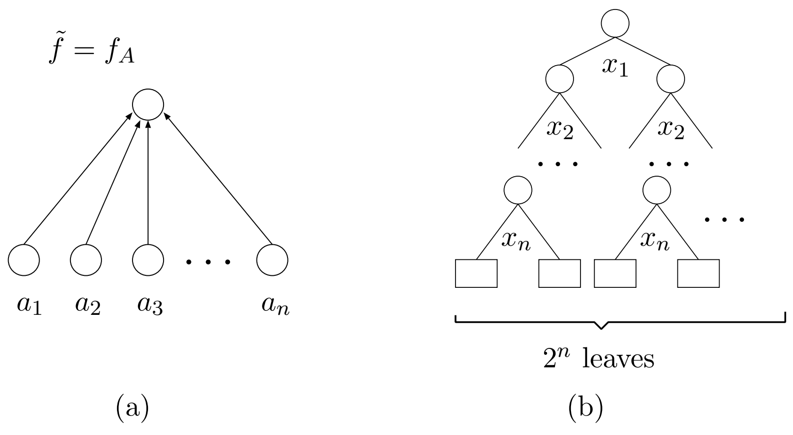


Figure 2.1: Structures of (a) $\tilde{f} = f_A$ and (b) $\tilde{t}$

of the same attribute in a path exactly move to the same child; the unvisited child and its descendants are redundant. It follows that $h(G_t) \leq n$ holds for any decision tree t.

2.2.2 Proof of Theorem 2.1

Let us introduce some notations. For any Boolean function $c : \mathbb{B}^n \to \mathbb{B}$, we write by $\mathcal{S}_R(c)$ the set of classifiers of R representing c.

Note that $\mathcal{S}_{\mathrm{ICF}}(c) \neq \emptyset$ and $\mathcal{S}_{\mathrm{DT}}(c) \neq \emptyset$ hold for any Boolean function $c : \mathbb{B}^n \to \mathbb{B}$. As a classifier representing c, we may compose a feature $\tilde{f}$ by taking $\tilde{f} = f_A$ and setting $f_A(x)$ to $c(x)$ for each $x \in \mathbb{B}^n$ (see Figure 2.1 (a)). Also, we may construct a decision tree $\tilde{t}$ representing c as follows: We take a binary rooted tree $G_{\tilde{t}} = (V_{\tilde{t}}, E_{\tilde{t}})$ such that the height is n and that all the inner nodes with height h are associated with attribute $(h + 1)$. (See Figure 2.1 (b).) Then $G_{\tilde{t}}$ has 2^n leaves and each leaf v corresponds to a binary vector $x \in \{0, 1\}^n$. For each leaf $v \in V_{\tilde{t}}^{\mathrm{leaf}}$, set $\ell_{\tilde{t}}(v)$ to $c(x)$ for the corresponding vector x. (Note that $\gamma_{\mathrm{ICF}}(\tilde{f}) = \gamma_{\mathrm{DT}}(\tilde{t}) = \Omega(2^n)$. Thus even if $\tilde{f}$ and $\tilde{t}$ are consistent classifiers, they are not classifiers of our interest.)

For the proof of the former part of Theorem 2.1 (i.e., *“if there exists a compact ε-decision tree, then there also exists a compact ε-feature”*), we show that, for any Boolean function c and any decision tree $t \in \mathcal{S}_{\mathrm{DT}}(c)$, there exists a feature $f \in \mathcal{S}_{\mathrm{ICF}}(c)$ such that $\gamma_{\mathrm{ICF}}(f) = O(\gamma_{\mathrm{DT}}(t))$. It is described as Lemma 2.1 and Corollary 2.1.

Lemma 2.1 *For any Boolean function $c : \mathbb{B}^n \to \mathbb{B}$ and any decision tree $t \in \mathcal{S}_{\mathrm{DT}}(c)$, there exists a feature $f \in \mathcal{S}_{\mathrm{ICF}}(c)$ such that $\gamma_{\mathrm{ICF}}(f) = O(\gamma_{\mathrm{DT}}(t))$.*

PROOF: Let t^v denote the subtree of $G_t = (V_t, E_t)$ which has node $v \in V_t$ as its root node, and let c^v denote a Boolean function represented by t^v (i.e., $t^v \in \mathcal{S}_{\mathrm{DT}}(c^v)$). For any node $v \in V_t$, we show that there exists a feature $f^v \in \mathcal{S}_{\mathrm{ICF}}(c^v)$ such that $\gamma_{\mathrm{ICF}}(f^v) \le C\gamma_{\mathrm{DT}}(t^v)$ for some constant $C > 0$. In this proof, we use $k_{\max} = 3$ for (2.1).

We prove this by an induction on the height h of node v. Assume $h = h(G_t)$. Then, v is a leaf and subtree t^v consists of one node v, which represents a constant function c^v that outputs $\ell_t(v)$ for any input. Then we compose the feature $f^v \in \mathcal{S}_{\mathrm{ICF}}(c^v)$ that has one fan-in (e.g., $f^v = f_{\{a_j\}}$, $\forall a_j \in A$) and that outputs $f^v(s) = \ell_t(v)$ for any input $s \in \mathbb{B}$. Then, $\gamma_{\mathrm{ICF}}(f^v) \le C \le C\gamma_{\mathrm{DT}}(t^v)$ for some constant $C > 0$.

Assume that the above statement holds for $h = h(G_t), h(G_t)-1, \ldots, h'$, and let $v \in V_t^{h'-1}$. If v is a leaf, we obtain $f^v \in \mathcal{S}_{\mathrm{ICF}}(c^v)$ such that $\gamma_{\mathrm{ICF}}(f^v) \le C\gamma_{\mathrm{DT}}(t^v)$ in the same way as above. Otherwise (i.e., if v is an inner node), subtree t^v represents a Boolean function $c^v = \bar{x}_j c^{v_{\mathrm{left}}} \vee x_j c^{v_{\mathrm{right}}}$, where $j = \ell_t(v) \in \{1, 2, \ldots, n\}$ is the index of an attribute. Then, we compose the feature $f^v = f_{\{a_j, f^{v_{\mathrm{left}}}, f^{v_{\mathrm{right}}}\}} = \bar{a}_j f^{v_{\mathrm{left}}} \vee a_j f^{v_{\mathrm{right}}}$. Since $f^{v_{\mathrm{left}}} \in \mathcal{S}_{\mathrm{ICF}}(c^{v_{\mathrm{left}}})$ and $f^{v_{\mathrm{right}}} \in \mathcal{S}_{\mathrm{ICF}}(c^{v_{\mathrm{right}}})$ hold from the assumption of the induction, and $a_j(x) = x_j$, $f^v \in \mathcal{S}_{\mathrm{ICF}}(c^v)$ holds.

Let us denote by $V'_{f^v} = V_{f^v} \setminus A$ the set of features which are not initial ones in V_{f^v}. From the above composition of f^v, $|V'_{f^v}| = |V_{t^v}|$ holds. We have

$$\begin{aligned}
\gamma_{\mathrm{ICF}}(f^v) &\le \gamma_{\mathrm{ICF}}(f^{v_{\mathrm{left}}}) + \gamma_{\mathrm{ICF}}(f^{v_{\mathrm{right}}}) + \log_2 k_{\max} + 2^3 + \log_2 \binom{|V_{f^v}|}{3} \\
&\quad + |V'_{f^v}|(\log_2 \binom{|V_{f^v}|}{3} - \log_2 \binom{|V_{f^v}|-1}{3}) \\
&\le \gamma_{\mathrm{ICF}}(f^{v_{\mathrm{left}}}) + \gamma_{\mathrm{ICF}}(f^{v_{\mathrm{right}}}) + \log_2 3 + 2^3 + C\log_2(|V_{t^v}| + n) \\
&\quad + |V_{t^v}| \log_2(1 + \frac{3}{|V_{f^v}| - 3}) \\
&\le C'\gamma_{\mathrm{DT}}(t^{v_{\mathrm{left}}}) + C'\gamma_{\mathrm{DT}}(t^{v_{\mathrm{right}}}) \\
&\le C''\gamma_{\mathrm{DT}}(t^v)
\end{aligned}$$

for some constants $C, C', C'' > 0$, which completes the induction. □

Let $c(x) = \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2\bar{x}_4 \vee x_1\bar{x}_3x_4 \vee x_1x_3$. Then the feature $f^4 \in \mathcal{S}_{\mathrm{ICF}}(c)$ in Figures 1.1 and 1.2 is composed by the above way from the decision tree $t \in \mathcal{S}_{\mathrm{DT}}(c)$ in Figures 1.4 and 1.5. E.g., for four nodes v_1 to v_4 of t, $f^{v_1} = f^4$, $f^{v_2} = f^3$, $f^{v_3} = f^2$ and $f^{v_4} = f^1$ are composed respectively. (Note that features for constant functions are omitted in Figure 1.1.)

From the above lemma, the following corollary is immediate.

Corollary 2.1 *Assume that there exists a Boolean function c such that $e(c, X) \leq \varepsilon$ holds. Then if there exists a compact ε-decision tree $t \in \mathcal{S}_{\mathrm{DT}}(c)$, there also exists a compact ε-feature $f \in \mathcal{S}_{\mathrm{ICF}}(c)$.*

Now let us go on to the proof of the latter part of Theorem 2.1 (i.e., *"the converse does not hold"*). We show that there exists a tuple (ε, X, y) such that there is a compact ε-feature but is no compact ε-decision tree. We observe four cases of (ε, X, y) (i.e., whether $X = \mathbb{B}^n$ or $X \subset \mathbb{B}^n$ and whether $\varepsilon = 0$ or $\varepsilon > 0$), each of which corresponds to one of the following lemmas.

Lemma 2.2 *Assume that we are given a set of data elements $X = \mathbb{B}^n$. Then there exists an oracle y on $\mathbb{B}^n$ such that there is a compact composed feature f consistent with $X = \mathbb{B}^n$ but there is no compact decision tree t consistent with $X = \mathbb{B}^n$.*

Proof: Let oracle y be a parity function such that

$$y(x) = \begin{cases} 1 & \text{if } \sum_{j=1}^{n} x_j \text{ is odd,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.3}$$

We can compose a required feature f as follows: For each attribute $j \in \{1, 2, \ldots, n-1\}$, we compose feature f^j as follows:

$$f^j = \begin{cases} f_{\{a_j, a_{j+1}\}} &= \bar{a}_j a_{j+1} \vee a_j \bar{a}_{j+1} & \text{if } j = 1, \\ f_{\{f^{j-1}, a_{j+1}\}} &= \bar{f}^{j-1} a_{j+1} \vee f^{j-1} \bar{a}_{j+1} & \text{otherwise.} \end{cases} \tag{2.4}$$

Clearly, $f^j(x) = 1$ if and only if $\sum_{j'=1}^{j+1} x_{j'}$ is odd; it follows that $e(f^{n-1}, X) = 0$. For $f = f^{n-1}$, the representation complexity $\gamma_{\mathrm{ICF}}(f)$ is determined as follows: Let us denote by $G_f = (V_f, E_f)$ the DAG embodying feature f. For each feature $f_S \in V_f \setminus A$, since the number of its fan-ins is 2, we use $k_{\max} = 2$ and $|S| = 2$ in (2.1). Then since $|V_f| = n + (n-1) = 2n - 1$, we observe $\gamma_{\mathrm{ICF}}(f) = O(n \log n)$.

We show that any consistent decision tree t requires $\gamma_{\mathrm{DT}}(t) = \Omega(2^n)$ for the representation complexity. For this, we show the following two propositions.

Proposition 2.1 *For any $J \subset \{1, 2, \ldots, n\}$ and any true example $x^1 \in X^1$, there exists a false example $x^0 \in X^0$ such that $x^1|_J = x^0|_J$.*

Proof: It is sufficient to show that such an x^0 exists for any $J = \{1, 2, \ldots, n\} \setminus \{j\}$, $j = 1, 2, \ldots, n$, and any $x^1 \in X^1$. Since we have all the n-dimensional vectors as $X = X^1 \cup X^0$, we find exactly one example $x^0 \in X^0$ satisfying $x^1|_J = x^0|_J$, by taking $x^0 = (x_1^1, x_2^1, \ldots, x_{j-1}^1, \bar{x}_j^1, x_{j+1}^1, \ldots, x_n^1)$. □

Proposition 2.2 *For any $J = \{1, 2, \ldots, n\} \setminus \{j\}$, $j = 1, 2, \ldots, n$, and any true example $x^1 \in X^1$, there exists no true example $x \in X^1$ ($x^1 \neq x$) such that $x^1|_J = x|_J$ holds.*

Proof: Assume that there exist two examples $x^1, x \in X^1$ such that $x^1|_J = x|_J$ for some $J = \{1, 2, \ldots, n\} \setminus \{j\}$. Since $x^1 \neq x$, $x^1_j \neq x_j$ holds; it follows that either $x^1 \in X^0$ or $x \in X^0$ holds as y is a parity function, which contradicts the assumption. □

Let us denote by $G_t = (V_t, E_t)$ the binary rooted tree of t. From Proposition 2.1, for each $x^1 \in X^1$, there exists an example $x^0 \in X^0$ which visits the same nodes as x^1 visits, in the depth of less than $n-1$. Since t is consistent and $h(G_t) \leq n$, x^1 and x^0 are separated at an inner node $v \in V_t^{\text{inner},n-1}$ in the depth of $n-1$. From Proposition 2.2, there is no $x \in X^1$ ($x^1 \neq x$) that visits v, and thus there is no example $x' \in X^0$ that visits v. Since there are $|X^1| = 2^n/2 = 2^{n-1}$ examples for x^1, $|V_t^{\text{inner},n-1}| = 2^{n-1}$ holds. Then the number of leaves $|V_t^{\text{leaf}}|$ equals $2 \cdot 2^{n-1} = 2^n$. Hence we have $\gamma_{\text{DT}}(t) = \Omega(2^n)$.
This proves Lemma 2.2. □

Lemma 2.3 *Assume that we are given a constant $k \in \{1, 2, \ldots, n\}$. Then there exist a set of $2^{n'}$ data elements X and an oracle y on $\mathbb{B}^n$, where we denote $n' = \lfloor n/k \rfloor$, such that there is a compact composed feature f consistent with $X \subseteq \mathbb{B}^n$ but there is no compact decision tree t consistent with $X \subseteq \mathbb{B}^n$.*

Proof: For the data set X, we associate each n'-dimensional vector z in $\mathbb{B}^{n'}$ with an example $x^z \in X$ as follows: we set $x^z_j = z_j$ for the first n' attribute values (i.e., $j = 1, 2, \ldots, n'$) and set $x^z_j = z_{j-n'\lfloor j/n' \rfloor}$ for the rest $n - n'$ attribute values (i.e., $j = n'+1, n'+2, \ldots, n$). For the oracle y, we take the parity function of the first n' attributes, as given in (2.3).

Clearly, we can compose the feature f from the first n' attributes such that $e(f, X) = 0$ and $\gamma_{\text{ICF}}(f) = O(n \log n)$ as in the proof for Lemma 2.2. On the other hand, for any Boolean function $c : \mathbb{B}^n \to \mathbb{B}$ with $e(c, X) = 0$, we see that there is no consistent decision tree t with $\gamma_{\text{DT}}(t) = o(2^n)$, since otherwise we would have obtained a decision tree t with $\gamma_{\text{DT}}(t) = o(2^n)$ in Lemma 2.2. □

Lemma 2.4 *Assume that we are given a set of data elements $X = \mathbb{B}^n$ and a constant $\varepsilon \in [0, 1/2)$. Then there exists an oracle y on $\mathbb{B}^n$ such that there is a compact composed ε-feature f but there is no compact ε-decision tree t.*

Proof: Let oracle y be the parity function as given in (2.3). From the proof for Lemma 2.2, there is a feature f such that $e(f, X) = 0 \leq \varepsilon$ and $\gamma_{\text{ICF}}(f) = O(n \log n)$.

For any ε-decision tree t, let us denote by $G_t = (V_t, E_t)$ the binary rooted tree of t and by $\ell_t : V_t \to \{1, 2, \dots, n\} \cup \mathbb{B}$ the label of t. We show that G_t contains at least $(1-2\varepsilon)|X| = (1-2\varepsilon)2^n$ leaves, by which $\gamma_{\mathrm{DT}}(t) = \Omega((1-2\varepsilon)2^n)$ follows. The error rate of t is given by

$$\begin{aligned} e(t, X) &= \frac{1}{|X|}\Big(\sum_{v \in V_t^{\mathrm{leaf}}:\ \ell_t(v)=1} |X^{v,0}| + \sum_{v \in V_t^{\mathrm{leaf}}:\ \ell_t(v)=0} |X^{v,1}|\Big) \\ &\geq \frac{1}{|X|}\Big(\sum_{v \in V_t^{\mathrm{leaf}} \setminus V_t^{\mathrm{leaf},n}:\ \ell_t(v)=1} |X^{v,0}| + \sum_{v \in V_t^{\mathrm{leaf}} \setminus V_t^{\mathrm{leaf},n}:\ \ell_t(v)=0} |X^{v,1}|\Big). \qquad (2.5) \end{aligned}$$

Proposition 2.3 *For each leaf $v \in V_t^{\mathrm{leaf}} \setminus V_t^{\mathrm{leaf},n}$, it holds $|X^{v,0}| = |X^{v,1}|$.*

PROOF: Let $v \in V_t^{\mathrm{leaf},h}$ be a leaf whose depth is $h(< n)$, and $J^w \subseteq \{1, 2, \dots, n\}$ denote the set of the indices of attributes used in the path between v and the root. Suppose that we repeat branching at v and its resulting children recursively, by the attributes $J = \{1, 2, \dots, n\} \setminus J^w$ until each of the resulting leaves has depth n. the parent v' of each of these leaves has depth $n-1$, we see that $|X^{v',1}| = |X^{v',0}| = 1$ holds from the proof of Lemma 2.2. This means that $|X^{v,1}| = |X^{v,0}|$ holds. $\square$

By this proposition, we see that (2.5) can be written as follows.

$$e(t, X) \geq \frac{1}{|X|} \sum_{v \in V_t^{\mathrm{leaf}} \setminus V_t^{\mathrm{leaf},n}} \frac{|X^v|}{2} = \frac{1}{2|X|}(|X| - |V_t^{\mathrm{leaf},n}|) = \frac{1}{2^{n+1}}(2^n - |V_t^{\mathrm{leaf},n}|).$$

Then since $e(t, X) \leq \varepsilon$, we have $|V_t^{\mathrm{leaf},n}| \geq 2^n - \varepsilon 2^{n+1}$, as required. $\square$

Lemma 2.5 *Assume that we are given constants $k \in \{1, 2, \dots, n\}$ and $\varepsilon \in [0, 1/2)$. Then there exist a set of $2^{n'}$ data elements X and an oracle y on $\mathbb{B}^n$, where we denote $n' = \lfloor n/k \rfloor$, such that there is a compact composed ε-feature f but there is no compact ε-decision tree t.*

PROOF: Let X be the set of examples given in Lemma 2.3, and oracle y be the parity function of the first n' attributes as given in (2.3). Clearly we can compose a consistent feature f from the first n' attributes such that $e(f, X) = 0 \leq \varepsilon$ and $\gamma_{\mathrm{ICF}}(f) = O(n \log n)$ as in the proof for Lemma 2.3.

Since $x_j = x_{j+n'} = x_{j+2n'} = \cdots = x_{j+(k-1)n'}$ $(j = 1, 2, \dots, n')$ for all examples $x \in X$, for any Boolean function $c : \mathbb{B}^n \to \mathbb{B}$, there exists an equivalent Boolean function $c' : \mathbb{B}^{n'} \to \mathbb{B}$ of the first n' attributes. Then it is sufficient to consider the training set $X' = \{0, 1\}^{n'}$ of $2^{n'}$ examples. From Lemma 2.4, any ε-decision tree t requires $\gamma_{\mathrm{DT}}(t) = \Omega((1-2\varepsilon)2^n)$ for the representation complexity. $\square$

Note that, in real situations, a feature f does not necessarily attain a small true error rate since f is a compact ε-feature. Also, it may be a hard problem to find a compact ε-feature for a given tuple (ε, X, y).

2.3 Experimental analysis on feature determination

In this section, we investigate whether we can construct good ICF classifiers in the sense of true error rate. The composition of a feature f_S, as defined in Definition 1.1, consists of two tasks: (1) to select a feature set S, and (2) to determine its Boolean function $f_S : \mathbb{B}^S \to \mathbb{B}$. The task (1) should be carried out with a greater care due to the largeness of the degree of freedom, while the function determination of the task (2) can be established naturally in view of learning from a data set. We introduce two schemes for function determination in Section 2.3.1: majorization and extended majorization.

Then we show how to examine the true error rates of features determined by the introduced schemes. We computationally generate a large amount of DAGs at random and determining the Boolean function of each node (i.e., a feature) by the schemes. A set of such generated features in this way may be an approximation to the entire set of features composed by the schemes. The minimum true error rate in the set suggests the potential performance of features composed by the schemes, in the sense that we *can* achieve that true error rate. Then we compare such potential performance of ICF with that of DT by experiments. (A set of many decision trees are generated in the similar way as features.)

The generation procedures for ICF and DT are described in Section 2.3.2, and we conduct the experimental studies on these procedures in Section 2.3.3. The computational results reveal that features composed by the introduced schemes have better potential than decision trees, and that extended majorization is better than majorization.

2.3.1 Feature determination schemes

Majorization.

Let us take a nonempty set S of features. For an $|S|$-dimensional vector $s \in \mathbb{B}^S$, let us define $X_{S,s} = \{x \in X \mid x|_S = s\}$, $X^1_{S,s} = X^1 \cap X_{S,s}$ and $X^0_{S,s} = X^0 \cap X_{S,s}$. For example, if we take $S = \{a_3, a_4\}$ and $s = (1, 0)$ in Table 1.1, then we have $X_{S,s} = \{x^2, x^3\} = \{(0, 1, 1, 0), (1, 0, 1, 0)\}$, $X^1_{S,s} = \{x^2\} = \{(0, 1, 1, 0)\}$ and $X^0_{S,s} = \{x^3\} = \{(1, 0, 1, 0)\}$. In majorization, we determine $f_S(s)$ by the distribution of classes in $X_{S,s}$ as follows:

$$f_S(s) = \begin{cases} 1 & \text{if } |X^1_{S,s}| > |X^0_{S,s}|, \\ 0 & \text{if } |X^1_{S,s}| < |X^0_{S,s}|, \\ b & \text{otherwise,} \end{cases} \tag{2.6}$$

where $b \in \mathbb{B}$ is a prescribed constant. E.g., for $S = \{a_3, a_4\}$ in Table 1.1 and $b = 0$, majorization sets $f_S(00) = f_S(01) = f_S(10) = 0$ and $f_S(11) = 1$. Thus f_S is a Boolean function $f_S(x) = x_3x_4$. Note that, if S is a singleton $S = \{g\}$, then f_S becomes equivalent with g (i.e., $f_S(x) = g(x)$ holds for all $x \in \mathbb{B}^n$).

For a given set $S = \{f^1, f^2, \dots, f^d\}$ of d features, it takes $O(d|X| + 2^d)$ time to compose the feature f_S by majorization: In order to count the numbers $|X^1_{S,s}|$ and $|X^0_{S,s}|$ for each $s \in \mathbb{B}^S$, we need to enumerate $x|_S = (f^1(x), f^2(x), \dots, f^d(x))$ for each $x \in X$, which requires $O(d|X|)$ time. (Here we assume that $f^j(x)$ for $j = 1, 2, \dots, d$ is computed in constant time.) In addition, we need to determine $f_S(s) \in \mathbb{B}$ for each $s \in \mathbb{B}^S$, which requires $O(2^d)$ time.

As to the empirical error rate of a feature composed by majorization, we derive the following proposition.

Proposition 2.4 *Assume that we are given a data set X labeled by the oracle y. Let us denote by S and S^+ sets of features such that $S \subseteq S^+$. Then two features f_S and f_{S^+} composed by majorization* (2.6) *satisfy $e(f_{S^+}, X) \le e(f_S, X)$.*

Proof:

$$\begin{aligned} e(f_S, X) &= \frac{1}{|X|} \sum_{s \in \mathbb{B}^S} \min\{|X^1_{S,s}|, |X^0_{S,s}|\} \\ &= \frac{1}{|X|} \sum_{s \in \mathbb{B}^S} \min\{ \sum_{s^+ \in \mathbb{B}^{S^+} : s^+|_S = s} |X^1_{S^+,s^+}|, \sum_{s^+ \in \mathbb{B}^{S^+} : s^+|_S = s} |X^0_{S^+,s^+}|\} \\ &\ge \frac{1}{|X|} \sum_{s \in \mathbb{B}^S} \sum_{s^+ \in \mathbb{B}^{S^+} : s^+|_S = s} \min\{|X^1_{S^+,s^+}|, |X^0_{S^+,s^+}|\} \\ &= \frac{1}{|X|} \sum_{s^+ \in \mathbb{B}^{S^+}} \min\{|X^1_{S^+,s^+}|, |X^0_{S^+,s^+}|\} \\ &= e(f_{S^+}, X). \end{aligned}$$

□

Note that the feature f_S composed by majorization attains the smallest empirical error rate among all possible functions from $\mathbb{B}^S$ to $\mathbb{B}$. (Note that $2^{2^{|S|}}$ functions are possible.) Then the following corollaries are immediate.

Corollary 2.2 *Let feature f_S be composed by majorization. Then $e(f_S, X) \leq e(f', X)$ holds for any $f' \in S$.*

Corollary 2.3 *Let feature f with DAG $G_f = (V_f, E_f)$ be composed by majorization. Then it holds $e(f, X) \leq e(f', X)$ for any $f' \in V_f$.*

From these corollaries, we may obtain features with small empirical error rates by iterative composition.

Now let us consider which feature set S results in a small empirical error rate $e(f_S, X)$. For two features f^1 and f^2, let us define the *difference between f^1 and f^2 on X*, denoted by $e_{\text{diff}}(f^1, f^2, X)$, as follows:

$$e_{\text{diff}}(f^1, f^2, X) = \frac{1}{|X|}|\{x \in X \mid f^1(x) \neq f^2(x)\}|.$$

Then the following proposition gives us a suggestion on the selection of S.

Proposition 2.5 *Assume that we are given a data set X and two features f^1 and f^2 composed by majorization* (2.6) *on X. Then the feature f_S with $S = \{f^1, f^2\}$ composed by majorization satisfies:*

$$e_{\text{diff}}(f^1, f^2, X) \geq \max\{e(f^1, X), e(f^2, X)\} - e(f_S, X).$$

Proof: Without loss of generality, we assume $e(f^1, X) \leq e(f^2, X)$ and $b = 0$. If $f_S(00) = 0$ and $f_S(11) = 1$ hold (i.e., $|X^1_{S,(00)}| \leq |X^0_{S,(00)}|$ and $|X^1_{S,(11)}| > |X^0_{S,(11)}|$), then we have

$$\begin{aligned}
e_{\text{diff}}(f^1, f^2, X) &= \frac{1}{|X|}(|X_{S,(01)}| + |X_{S,(10)}|) \\
&= \frac{1}{|X|} \sum_{s \in \{(01),(10)\}} (\max\{|X^1_{S,s}|, |X^0_{S,s}|\} + \min\{|X^1_{S,s}|, |X^0_{S,s}|\}) \\
&\geq \frac{1}{|X|} \sum_{s \in \{(01),(10)\}} (\max\{|X^1_{S,s}|, |X^0_{S,s}|\} - \min\{|X^1_{S,s}|, |X^0_{S,s}|\}) \\
&= \frac{1}{|X|}[(|X^1_{S,(00)}| + |X^0_{S,(11)}| + \sum_{s \in \{(01),(10)\}} \max\{|X^1_{S,s}|, |X^0_{S,s}|\}) \\
&\quad - (|X^1_{S,(00)}| + |X^0_{S,(11)}| + \sum_{s \in \{(01),(10)\}} \min\{|X^1_{S,s}|, |X^0_{S,s}|\})] \\
&\geq \frac{1}{|X|}(|X^1_{S,(00)}| + |X^0_{S,(11)}| + |X^0_{S,(01)}| + |X^1_{S,(10)}|) - e(f_S, X) \\
&= e(f^2, X) - e(f_S, X),
\end{aligned}$$

and thus $e_{\mathrm{diff}}(f^1, f^2, X) \geq e(f^1, X) - e(f_S, X)$ also holds from the assumption $e(f^1, X) \leq e(f^2, X)$.

On the other hand, assume that $f_S(00) = 1$ or $f_S(11) = 0$ hold (i.e., $|X^1_{S,(00)}| > |X^0_{S,(00)}|$ or $|X^1_{S,(11)}| \leq |X^0_{S,(11)}|$). Consider the case of $f_S(00) = 1$. Then since f^1 and f^2 are composed by majorization, $f_S(01) = f_S(10) = 0$ and $f_S(11) = 1$ hold (since if $f_S(01) = 1$, for example, then it contradicts that f^1 is composed by majorization). Also we have

$$|X^1_{S,(00)}| + |X^1_{S,(01)}| \leq |X^0_{S,(00)}| + |X^0_{S,(01)}|, \quad |X^1_{S,(00)}| + |X^1_{S,(10)}| \leq |X^0_{S,(00)}| + |X^0_{S,(10)}|,$$

and thus

$$|X^1_{S,(00)}| - |X^0_{S,(00)}| \leq |X^0_{S,(01)}| - |X^1_{S,(01)}|, \quad |X^1_{S,(00)}| - |X^0_{S,(00)}| \leq |X^0_{S,(10)}| - |X^1_{S,(10)}|,$$

respectively. Thus we have $|X^1_{S,(00)}| - |X^0_{S,(00)}| \leq \min\{|X_{S,(01)}|, |X_{S,(10)}|\}$. Then

$$\begin{aligned} e(f^2, X) - e(f_S, X) &= (|X^1_{S,(00)}| - |X^0_{S,(00)}|) + (|X^0_{S,(01)}| - |X^1_{S,(01)}|) \\ &\leq \min\{|X_{S,(01)}|, |X_{S,(10)}|\} + |X_{S,(01)}| \\ &\leq e_{\mathrm{diff}}(f^1, f^2, X). \end{aligned}$$

For the case of $f_S(11) = 0$, the proof is analogous with the above. $\square$

The determination of difference $e_{\mathrm{diff}}(f^1, f^2, X)$ takes $O(|X|)$ time, which is not much smaller than the time for composing feature $f_{\{f^1, f^2\}}$ by majorization. No method for checking whether $e(f_S, X) < e(f^1, X), e(f^2, X)$ holds without composing f_S itself is known.

Extended majorization.

The above majorization is regarded as an application of the concept of *Bayes Rule*, well-known in statistics [67, 81]: The framework of Bayes Rule assumes that data elements are identically and independently distributed under a certain probability distribution. For a nonempty set S of features and a vector $s \in \mathbb{B}^S$, we write by $P^1_{S,s}$ (resp., $P^0_{S,s}$) the posterior probability of $y(x) = 1$ (resp., 0) under $x \in X_{S,s}$. (Note that $P^1_{S,s} + P^0_{S,s} = 1$ holds.) Bayes Rule sets $f_S(s) = 1$ (resp., 0) if $P^1_{S,s} > P^0_{S,s}$ (resp., $P^1_{S,s} < P^0_{S,s}$) holds and $f_S(s) = b$ for a constant $b \in \mathbb{B}$ otherwise, by which f_S attains the smallest true error rate among all possible functions from $\mathbb{B}^S$ to $\mathbb{B}$.

However, the probabilities $P^1_{S,s}$ and $P^0_{S,s}$ are usually unavailable to us. Majorization is then an approximation to Bayes Rule in the sense that, if $X_{S,s} \neq \emptyset$, it utilizes $|X^1_{S,s}|/|X_{S,s}|$ (resp., $|X^0_{S,s}|/|X_{S,s}|$) in place of $P^1_{S,s}$ (resp., $P^0_{S,s}$); otherwise, it sets $f_S(s) = b$.

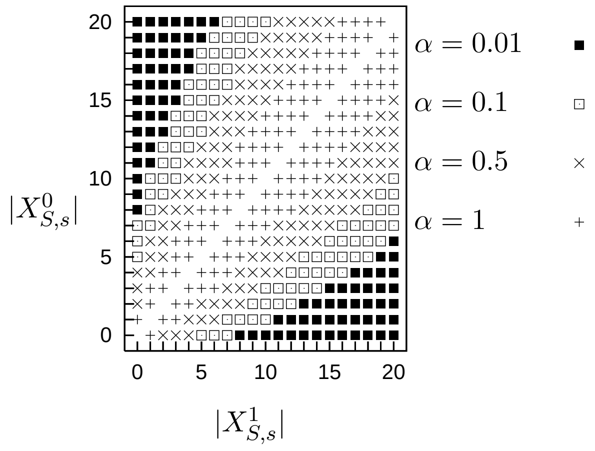


Figure 2.2: The area of $(|X^1_{S,s}|, |X^0_{S,s}|)$ on which the hypothesis is rejected

We consider that the classification $f_S(s) \in \mathbb{B}$ on an input $s \in \mathbb{B}^S$ is uncertain if $|X^1_{S,s}| \simeq |X^0_{S,s}|$ holds. In order to represent the state of such uncertainty, we introduce the nominal $*$, a missing bit.

Let us denote $\mathbb{M} = \mathbb{B} \cup \{*\}$. We extend feature f_S to a function that deals with $*$. The extended feature is denoted by a function $f^*_S : \mathbb{M}^S \to \mathbb{M}$, where S denotes a set of (extended) features.

The extended majorization is a scheme that determines the function f^*_S. In the extended majorization, for each $s \in \mathbb{M}^S$, the output $f^*_S(s)$ is determined based on the following statistical test [44]: The hypothesis is that true and false examples in $X_{S,s}$ are generated with the same probability (i.e., $P^1_{S,s} = P^0_{S,s} = 1/2$). If the hypothesis is accepted, we determine $f^*_S(s) = *$ (i.e., we cannot see the bias of classes). Otherwise, we determine $f^*_S(s) = 1$ or 0 by the major class in $X_{S,s}$. We use a parameter $\alpha \in [0, 1]$ to determine the rejection rate of the statistical test; if α is large (resp., small), then $f^*_S(s)$ is more likely to be 1 or 0 (resp., $*$). This is summarized as follows.

$$f^*_S(s) = \begin{cases} * & \text{if } H(M,m) > \alpha/2, \\ 1 & \text{if } H(M,m) \le \alpha/2 \text{ and } |X^1_{S,s}| > |X^0_{S,s}|, \\ 0 & \text{if } H(M,m) \le \alpha/2 \text{ and } |X^1_{S,s}| < |X^0_{S,s}|, \end{cases} \tag{2.7}$$

where $M = |X^1_{S,s}| + |X^0_{S,s}|$, $m = \min\{|X^1_{S,s}|, |X^0_{S,s}|\}$, and

$$H(M,m) = \sum_{i=0}^{m} \binom{M}{i} (\frac{1}{2})^i (\frac{1}{2})^{M-i} = \frac{1}{2^M} \sum_{i=0}^{m} \binom{M}{i}. \tag{2.8}$$

Figure 2.2 illustrates the area of $(|X^1_{S,s}|, |X^0_{S,s}|)$ on which the hypothesis is rejected (i.e.,

$f^*_S(s)$ is either 1 or 0) for $\alpha \in \{0.01, 0.1, 0.5, 1\}$. If the hypothesis on $(|X^1_{S,s}|, |X^0_{S,s}|)$ is rejected for an α_0, then it is also rejected for any $\alpha \geq \alpha_0$. Note that $H(M, m) > 1/2$ holds if and only if $|X^1_{S,s}| = |X^0_{S,s}|$ (i.e., $M = 2m$), and in this case, the hypothesis is always accepted for every value of $\alpha \in [0, 1]$. As derived from (2.7) and (2.8), $f^*_S(s) = *$ holds for any $s \in \mathbb{M}^S$ such that $|X_{S,s}| < 1 - \log_2 \alpha$ holds.

Hence the value of α may be determined according to the reliability of the data set; if the data set is not reliable due to such reasons as its smallness, noise, etc., then α should be taken small. On the other hand, if it is reliable, then α should be taken large.

Analogously with (2.1), the representation complexity for extended feature f^*, which we denote by $\gamma_{\mathrm{ICF}^*}(f^*)$, is defined as follows:

$$\gamma_{\mathrm{ICF}^*}(f^*) = \sum_{f^*_S \in V_{f^*} \setminus A} \left(\log_2 k_{\max} + 3^{|S|} \log_2 3 + \log_2 \binom{|V_{f^*}|}{|S|} \right). \tag{2.9}$$

In order to use $f^*_S : \mathbb{M}^S \to \mathbb{M}$ as a classifier, we compose a feature $f_S : \mathbb{M}^S \to \mathbb{B}$, where $f_S(s)$ is determined by majorization (2.6) for each input $s \in \mathbb{M}^S$. Then f_S becomes a Boolean function on $\mathbb{B}^n$, although V_{f_S} may contain some extended features.

We note that an extended feature with output $*$ represents *partial knowledge*, in the sense that it tells the bias of classes not for the entire data space but only for its subspace. Use of the uncertainty value $*$ is not new in computational learning theory (particularly in pattern recognition). If it is necessary to avoid the classification error, we may allow a classifier to output $*$, meaning "I cannot decide." This classification strategy is called a *reject option*, and has been studied in [25, 34, 58], for example. From a theoretical viewpoint of Boolean functions, a function $\mathbb{M}^n \to \mathbb{M}$ has been studied intensively under the framework of partially defined Boolean functions and LAD (e.g., [20, 21]).

2.3.2 Generation procedures for classifiers.

In the experiments of the next subsection, we approximate the entire set of features composed by majorization and extended majorization, and that of decision trees by generating a large amount of classifiers. For this, we introduce generation procedures both for ICF and for DT. Note that we distinguish a generation procedure from a construction algorithm, which outputs exactly one classifier.

ICF.

We introduce the following generation procedure GEN-ICF to compose many features by majorization. In GEN-ICF, F_h denotes a set of features whose heights are h. We compose

100 features for each F_h ($h = 1, 2, \ldots, 10$). This number of features is considered large enough to approximate the entire set of features from our preliminary experiments.

Procedure GEN-ICF

Input: A data set X with n attributes (and thus with the set of n initial features $A = \{a_1, a_2, \ldots, a_n\}$), an upper bound on the number of fan-ins $k_{\max}$.

Output: A set F of features.

```
Step 1: $F := F_0 := A$.
Step 2: For $h = 1, 2, \ldots, 10$:
    Step 2-1: $F_h := \emptyset$. Repeat the following 100 times:
        Step 2-1-1: Compose a feature as follows: Select $S \subseteq F$ at random so that
            $2 \leq |S| \leq k_{\max}$ and $S \cap F_{h-1} \neq \emptyset$. Determine $f_S$ by the majorization.
        Step 2-1-2: $F_h := F_h \cup \{f_S\}$.
    Step 2-2: $F := F \cup F_h$.
Step 3: Output $F$ and halt.
```

In the above procedure, the extended majorization can be used in Step 2-1-1 for feature determination, in place of majorization. We denote by GEN-ICF* such modified procedure and adopt it as the generation procedure to compose many features by extended majorization.

DT.

For DT, we utilize a construction algorithm C4.5 [65] for our generation procedure of decision trees. C4.5 has several adjustable parameters and options, and we generate many decision trees by determining their values at random. In the experiments, we generate 1000 decision trees for given X, which is considered large enough from our preliminary experiments. We show the generation procedure of decision trees as follows.

Procedure GEN-DT

Input: A data set X with n attributes.

Output: A set T of decision trees.

Step 1: Construct 1000 decision trees by applying C4.5 on X with randomly chosen parameter values and options as follows. We denote by T the set of constructed decision trees.

Criterion: In a decision tree t with $G_t = (V_t, E_t)$ under construction, the leaf $v \in V_t^{\text{leaf}}$ to be branched and its labeled attribute are selected based on one of two entropy-based criteria: *gain* and *gain ratio*. One of them are selected at random.

Weight: C4.5 does not branch a leaf v such that the number of examples visiting v (i.e., $|X^v|$) is not more than the specified value. We specify the value by selecting from $\{1, 2, \ldots, C\}$ at random. (Since we have observed that the result does not change to a great extent if $C \geq \lfloor 0.1|X| \rfloor$, we use $C = \lfloor 0.1|X| \rfloor$.)

CF: CF stands for *confidence level*, represented by percentage, at which rate pruning is performed during construction of a decision tree. We select its value from $\{1\%, 2\%, \ldots, 100\%\}$ at random. A small (resp., large) value of CF tends to produce a small (resp., large) decision tree.

Windowing: *Windowing* is such a process described as follows: Several decision trees are generated by choosing a subset $X' \subseteq X$ at random, from which a decision tree t is constructed. The true error rate of each t is estimated by $e(t, X \setminus X')$. Then the one attaining the smallest estimation is adopted as the output decision tree.

Whether windowing is applied or not is selected at random. If applied, the number of iteration is selected from $\{2, 3, \ldots, 10\}$ at random (10 times are large enough in our experience), and the size of X' is selected from $\{1, 2, \ldots, \lfloor 0.5|X| \rfloor\}$ at random (no significant change has been observed in our preliminary experiments if $|X'| \geq \lfloor 0.5|X| \rfloor$).

Step 2: Output T and halt.

Among the parameters and options described above, we have observed that CF has the largest influence on the performance of resulting decision trees.

2.3.3 Experimental studies

Experimental setting.

We use benchmark data sets from UCI Repository of Machine Learning Databases [42] in our experiments. In each data set Ω, an example $\omega \in \Omega$ is described by numerical and/or categorical attributes. Let N_{num} (resp., N_{cat}) denote the number of numerical (resp., categorical) attributes of Ω and $N = N_{\text{num}} + N_{\text{cat}}$. The summary of data sets is given in Table

Table 2.1: Data sets from UCI Repository of Machine Learning

Data		$\lvert\Omega\rvert$	$(\lvert\Omega^1\rvert, \lvert\Omega^0\rvert)$	N	$(N_{\text{num}}, N_{\text{cat}})$	n
AUS	(Australian credit approval)	690	(307,383)	14	(6,8)	18
BCW	(Breast cancer wisconsin)	683	(239,444)	9	(9,0)	12
BUPA	(BUPA liver disorders)	345	(200,145)	6	(6,0)	20
CAR	(Car evaluation)	1728	(518,1210)	6	(0,6)	12
CRX	(Credit approval)	653	(296,357)	15	(6,9)	15
FLAG	(Flag design)	194	(100,94)	28	(10,18)	10
HABER	(Haberman's survival)	294	(75,219)	3	(3,0)	31
HEART	(Heart disease)	270	(120,150)	13	(7,6)	12
IONO	(Ionosphere)	351	(225,126)	34	(34,0)	11
MUSH	(Mushroom)	8124	(3916,4208)	22	(0,22)	5
PIMA	(Pima indian diabetes)	768	(268,500)	8	(8,0)	22
TTT	(Tic-Tac-Toe endgame)	958	(626,332)	9	(0,9)	12
VOTES	(1984 US congressional votes)	435	(267,168)	16	(0,16)	11

2.1, where the rightmost column for n will be described later. E.g., each example in BCW corresponds to a patient, and its class represents whether it is malignant (true) or benign (false) to breast cancer. Attributes describe such information as clump thickness, uniformity of cell size, and so forth.

We make some modifications on the data sets as follows.

- Some data sets contain examples with missing attribute values (e.g., in BCW and CRX), and we exclude such examples.
- An example of CAR has one of {`unacc`,`acc`,`good`,`v-good`} for its class. We regard the examples with class `unacc` as false ones and the rest as true ones.
- FLAG has no class information, and thus we utilize an attribute "religion" as the class; if the attribute value is `Catholic`, then we regard the example as false. Otherwise, it is treated as a true example.
- Let us call examples having equivalent vectors but different classes *contradicting examples*. HABER contains such contradicting examples, and we exclude such examples.

The summary in Table 2.1 is based on the modified data sets as above.

Also, in order to treat a numerical and/or categorical data set Ω in our formulation, we need to transform Ω into a binary data set X. For this, we apply a discretization scheme

	$k_{\max} = 3$	$k_{\max} = 6$
GEN-ICF (majorization)	—□—	- - □ - -
GEN-ICF* (extended majorization)	—+—	- - + - -

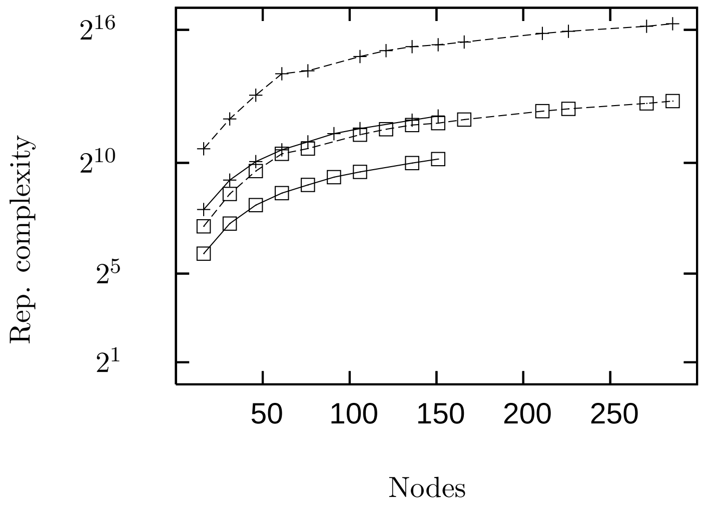


Figure 2.3: Representation complexity of generated features on BCW

named space based construction (SC) (with parameter values $\Gamma_{\text{SC}} = \Gamma_{\text{SC,PAIR}}$ and $\mathbb{V} = \mathbb{B}$), which was originally proposed by Mii [55] and will be studied in Chapter 4. This scheme guarantees that $X^1 \cap X^0 = \emptyset$ holds for the obtained binary data set X. The rightmost column for n of Table 2.1 shows the number of attributes of X.

We wish to estimate the true error rate of a classifier r. For this, we apply hold-out method [81]: We divide a binary data set X into halves at random, one for the *training set* X_{train} and the other for the *test set* X_{test}. We then generate classifiers from X_{train} by a generation procedure, where the true error rate of r is estimated by the error rate on the test set, i.e., $e(r, X_{\text{test}})$.

For GEN-ICF and GEN-ICF*, we use $k_{\max} \in \{3, 4, \ldots, 8\}$. Note that a large (resp., small) $k_{\max}$ result features with large (resp., small) representation complexity. For GEN-ICF, we use the constant $b = 0$ in (2.6). For GEN-ICF*, we use the parameter value $\alpha = 0.1$ in (2.7).

Results and discussion.

Let us take a classifier r generated from a representation model R (where R denotes either ICF or DT and r does either an ICF classifier or a decision tree, respectively). We observe the

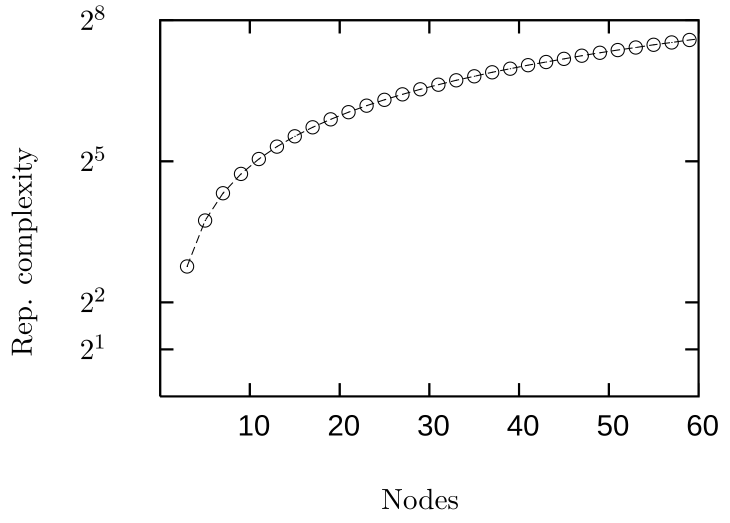


Figure 2.4: Representation complexity of generated decision trees on BCW

representation complexity $\gamma_R(r)$. As seen from (2.1), (2.2) and (2.9), $\gamma_R(r)$ is proportional to the number of nodes in the graph representing the structure of r. We show this relationship between the number of nodes and representation complexity in Figures 2.3 and 2.4, corresponding to ICF and DT, respectively. In these figures, the horizontal (resp., vertical) axis is for the number of nodes (resp., representation complexity), where the vertical axis is in the logarithmic scale. The representation complexity is taken as the average among the classifiers generated on BCW consisting of the same number of nodes. In Figure 2.3, we show the representation complexity of only $k_{\max} = 3$ and 6 for legibility.

We next observe the attained empirical and true error rates. We show the results of generation procedures GEN-ICF, GEN-ICF* and GEN-DT on BCW (resp., CAR) data set in Figures 2.5 to 2.7 (resp., Figures 2.8 to 2.10) respectively, where we use $k_{\max} = 3$. In these figures, for four segments of the number of nodes (which is proportional to representation complexity), a classifier is plotted by empirical error rate (horizontal axis) and true error rate (vertical axis). One can observe that a complex classifier is likely to attain a small empirical error rate, where this does not always hold for the case of true error rate.

A set of classifiers generated by GEN-ICF (resp., GEN-ICF* and GEN-DT) approximates the entire set of features composed by majorization (resp., features composed by extended majorization and decision trees). Then the best (resp., smallest) true error rate in a classifier set may indicate its potential performance. Let us call the best true error rate in a classifier set its *potential error rate*. As shown in the figures, the potential error rates of the three

Table 2.2: Potential error rates ($\times 10^2$) of GEN-ICF and GEN-DT

Data	GEN-ICF						GEN-DT
	$k_{\max} = 3$	4	5	6	7	8	
AUS	13.90	**13.06**	**12.45**	**12.75**	**13.15**	13.73	13.47
BCW	**2.36**	**2.66**	**2.54**	**2.86**	**2.92**	**2.83**	3.56
BUPA	29.19	**27.39**	**28.26**	**28.32**	**29.07**	29.36	29.19
CAR	5.52	3.18	2.15	2.27	2.23	2.49	0.92
CRX	**11.80**	**11.56**	**11.31**	**11.58**	**11.74**	12.01	11.83
FLAG	**9.17**	**8.45**	**8.96**	**9.79**	**9.07**	**9.58**	9.89
HABER	**21.63**	**21.70**	**21.76**	**21.70**	**21.90**	**21.70**	23.46
HEART	**14.37**	**13.25**	**13.7**	14.66	**14.51**	15.25	14.66
IONO	**7.78**	**7.78**	**7.78**	**7.95**	**8.29**	**8.18**	9.26
MUSH	0.17	0.00	0.00	0.00	0.00	0.00	0.00
PIMA	22.37	**22.03**	22.26	22.78	22.47	22.96	22.18
TTT	21.12	16.57	12.19	8.35	8.39	8.56	2.12
VOTES	**3.16**	**2.75**	**3.25**	**3.02**	**3.16**	**3.11**	3.94
POTENTIAL	12.50	11.56	11.27	11.23	11.30	11.52	11.11

procedures do not differ on BCW, while GEN-DT is best of all on CAR in terms of potential error rate; GEN-DT generates a classifier whose potential error rate is nearly 0.

Let us examine the potential error rates for all data sets. To compare the potential error rates of the three procedures, we iterate division of the data set X into X_{train} and X_{test} 10 times and take the average of potential error rates as the evaluator of a generation procedure. Tables 2.2 and 2.3 show the results of GEN-ICF and GEN-ICF* respectively, where the result of GEN-DT is present in both tables for convenience. A boldfaced error rate on ICF procedures indicates a value smaller than GEN-DT. A value at the bottom row (i.e., POTENTIAL) is the average of these potential error rates for all data sets.

ICF procedures outperform GEN-DT for all data sets except CAR and TTT. Also, as is observed from the bottom row, GEN-ICF* appears to be better than GEN-ICF; extended majorization can produce better features than majorization. The main reason for this is that a feature composed by extended majorization has $\mathbb{M}$-valued functions as its intermediate features, and it can produce more various Boolean functions than one composed by majorization. We observed that a larger value of $k_{\max}$ leads to a smaller potential error rate in general; a large $k_{\max}$ may produce features with complex structure more often than a small k_{max}

Table 2.3: Potential error rates ($\times 10^2$) of GEN-ICF* and GEN-DT

Data	GEN-ICF*						GEN-DT
	$k_{\max} = 3$	4	5	6	7	8	
AUS	**12.28**	**11.93**	**12.63**	**12.80**	**12.69**	**12.77**	13.47
BCW	**2.39**	**2.39**	**2.45**	**2.39**	**2.54**	**2.36**	3.56
BUPA	**28.09**	**27.34**	**27.28**	**28.03**	**27.80**	**27.91**	29.19
CAR	4.16	2.75	2.38	2.18	2.16	2.67	0.92
CRX	**11.25**	**11.34**	**11.10**	**11.52**	**11.46**	**11.52**	11.83
FLAG	**7.94**	**7.42**	**7.83**	**7.83**	**8.24**	**8.66**	9.89
HABER	**21.15**	**21.22**	**21.29**	**21.15**	**21.29**	**21.08**	23.46
HEART	**13.40**	**13.62**	**13.62**	**14.14**	**14.51**	**14.44**	14.66
IONO	**7.66**	**8.01**	**7.95**	**8.12**	**8.69**	**7.83**	9.26
MUSH	0.07	0.00	0.00	0.00	0.00	0.00	0.00
PIMA	22.42	**22.10**	**21.82**	22.94	22.71	22.29	22.18
TTT	17.66	14.73	10.41	9.95	9.25	8.95	2.12
VOTES	**2.56**	**2.65**	**2.93**	**2.93**	**2.93**	**2.84**	3.94
POTENTIAL	11.61	11.19	**10.89**	**11.07**	**11.09**	**11.02**	11.11

does, and thus various Boolean functions can be generated by a large $k_{\max}$. These results show that extended majorization generates potentially good ICF classifiers since there exists a good classifier in the feature set generated by GEN-ICF*. In other words, we *can* compose good features by extended majorization.

Then let us consider how to design a construction algorithm Λ that outputs *one* good ICF classifier. We may consider such Λ as a procedure that generates a (reasonably) large number of features similarly to the procedures in the experiments, and adopts one of them as the classifier by using some criteria. Since we do not know the true error rate of a classifier in the generation process of Λ, the criteria for adoption should reflect the tendency of true error rate to some extent (at least in the generated feature set).

Besides, Λ may include several parameters for regulation, similarly to many of construction algorithms proposed so far in previous studies. It is desirable for Λ to produce a good classifier when its parameter values are finely tuned up, but determining appropriate parameter values is usually difficult. Then what is more significant as a property of Λ is that a good classifier is produced for almost all parameter values.

Based on this observation, we will propose construction algorithms of ICF classifiers in the next Chapter 3.

2.4 Concluding remarks

In this chapter, we compared the potential performance between two representation models, ICF and DT. In Section 2.2, we showed that ICF is superior to DT in view of Occam's Razor, as is summarized in Theorem 2.1. In Section 2.3, we conducted experimental studies on two feature determination schemes, majorization and extended majorization. The results showed that these schemes enable us to construct better classifiers than decision trees, and that the latter can produce better ICF classifiers than the former.

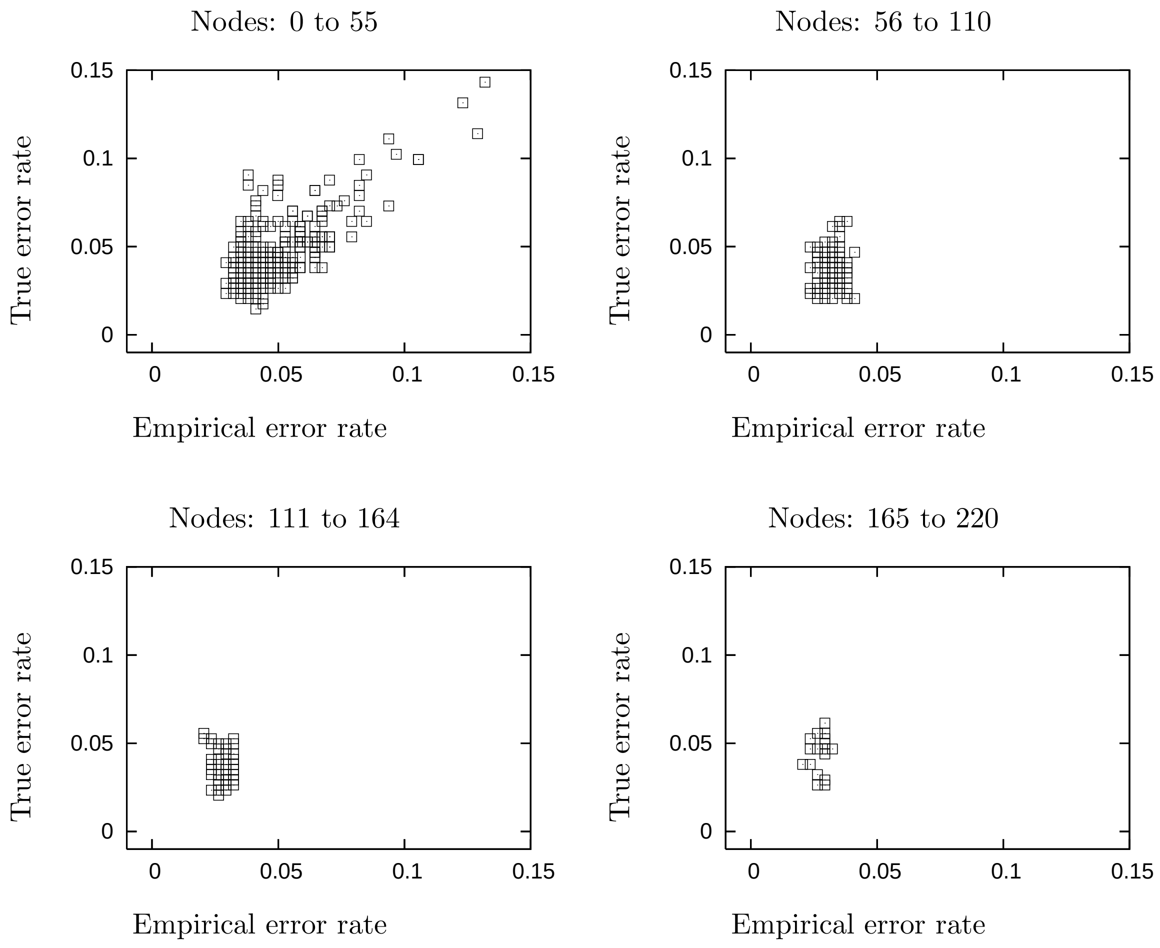


Figure 2.5: Spreads of features by GEN-ICF with $k_{\max} = 3$ on BCW

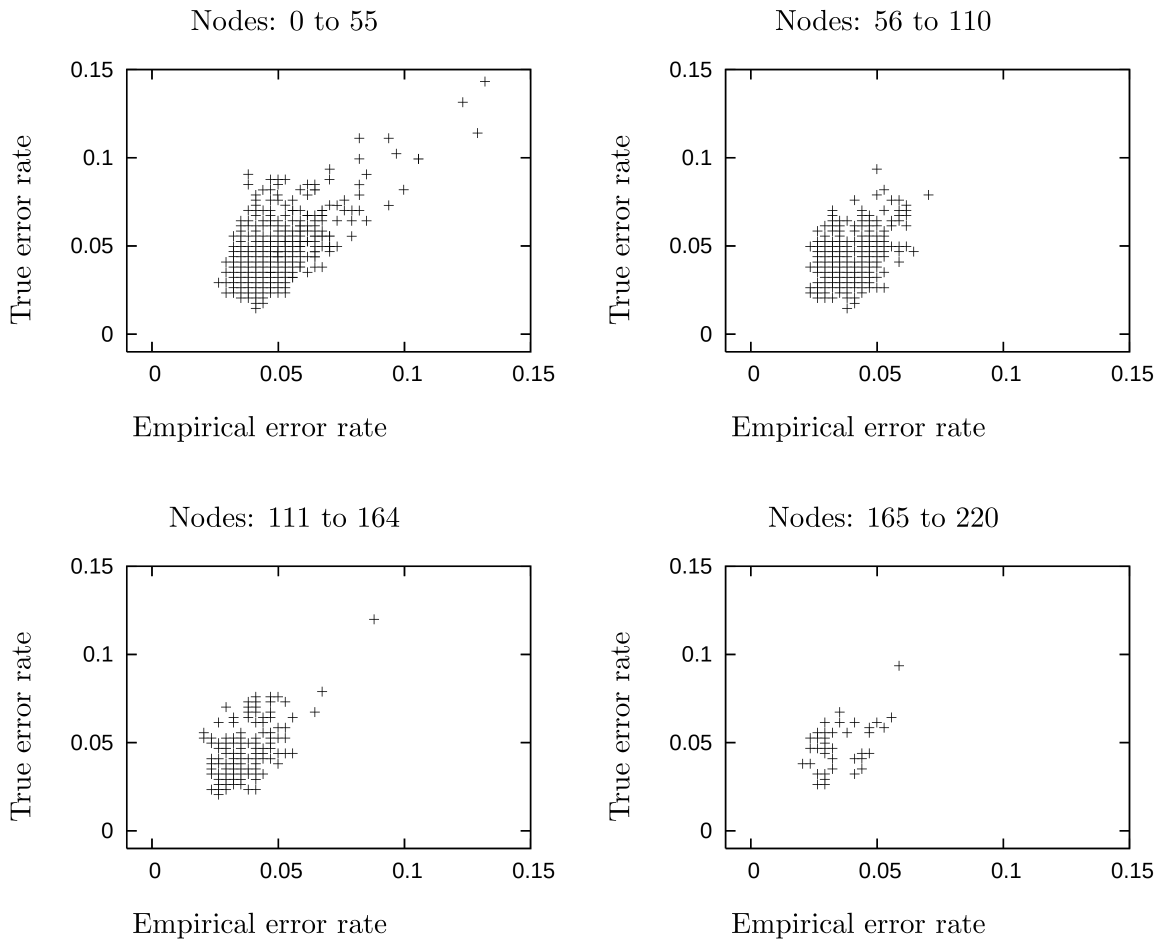


Figure 2.6: Spreads of features by GEN-ICF* with $k_{\max} = 3$ on BCW

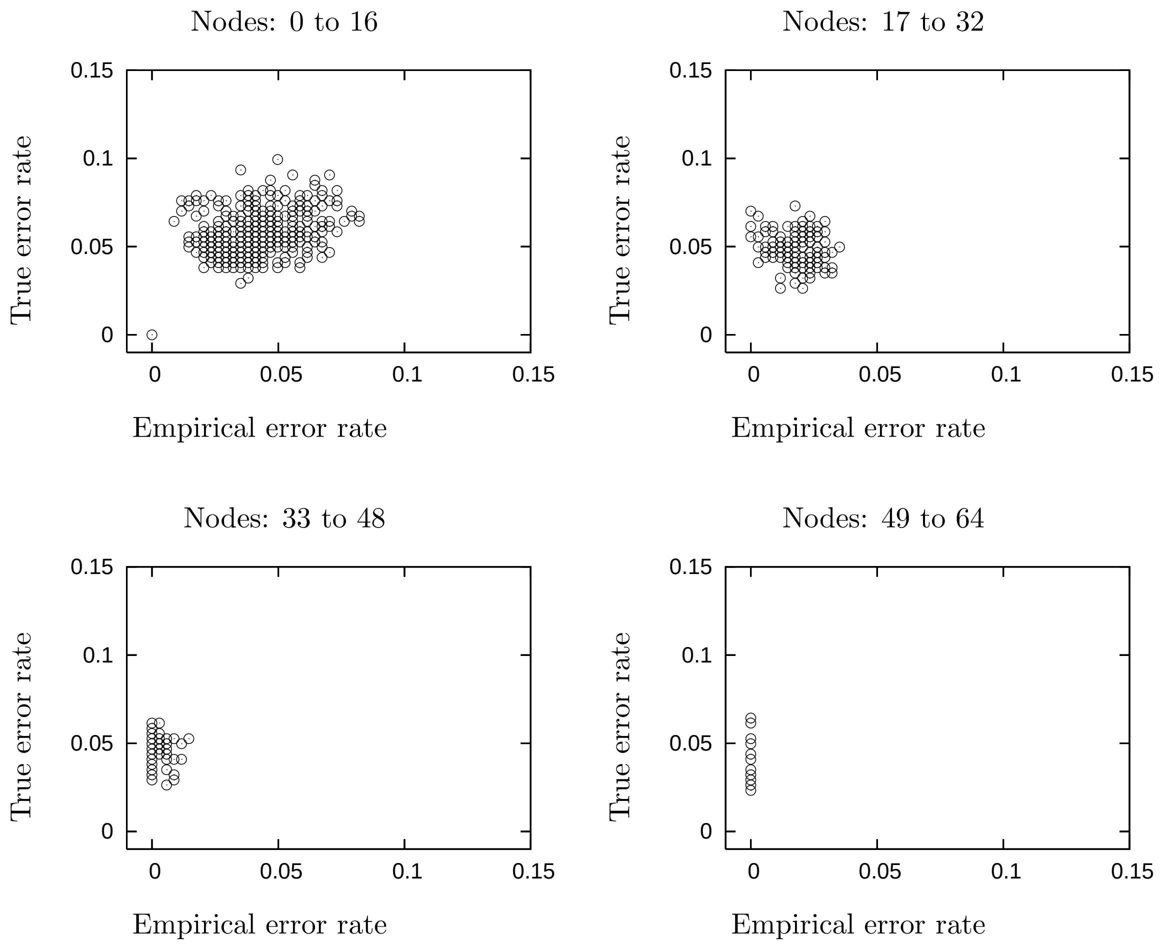


Figure 2.7: Spreads of decision trees by GEN-DT on BCW

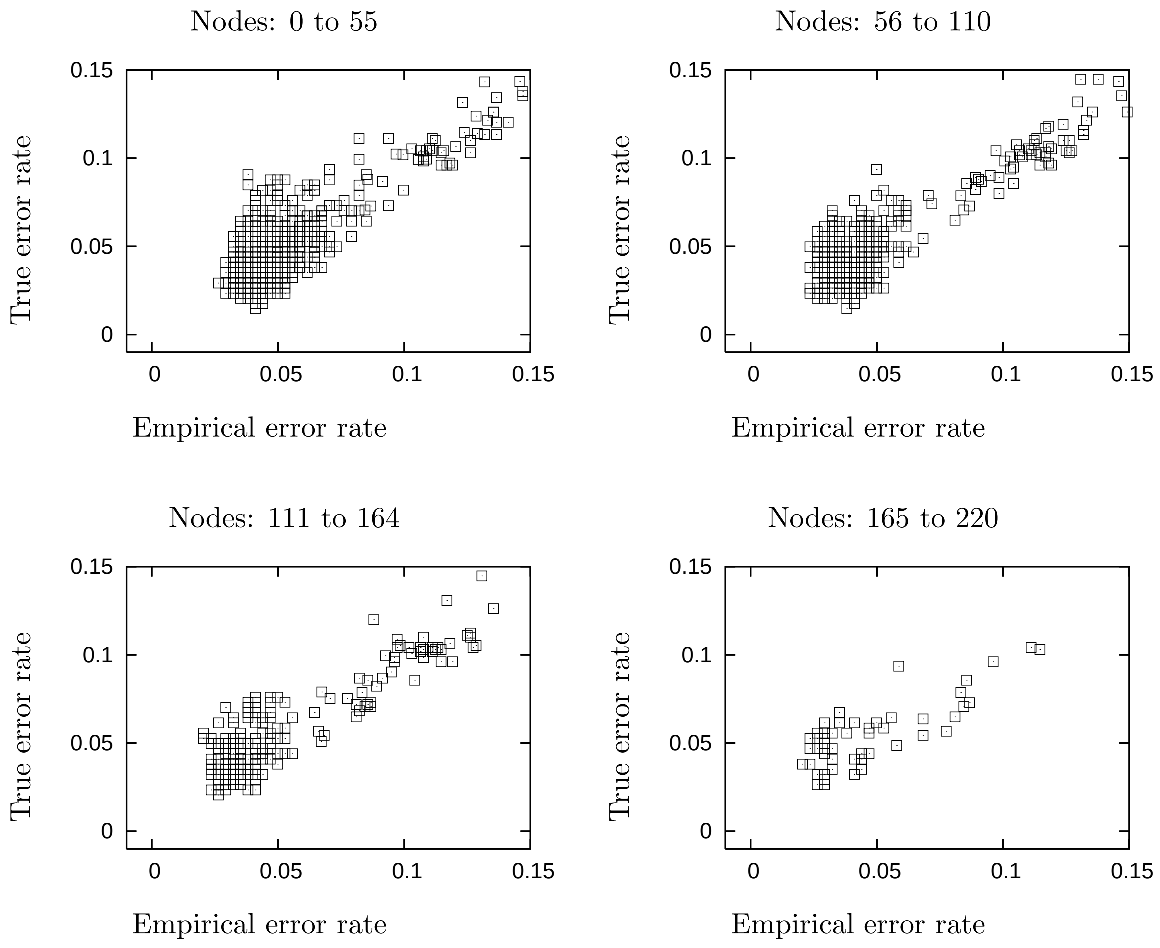


Figure 2.8: Spreads of features by GEN-ICF with $k_{\text{max}} = 3$ on CAR

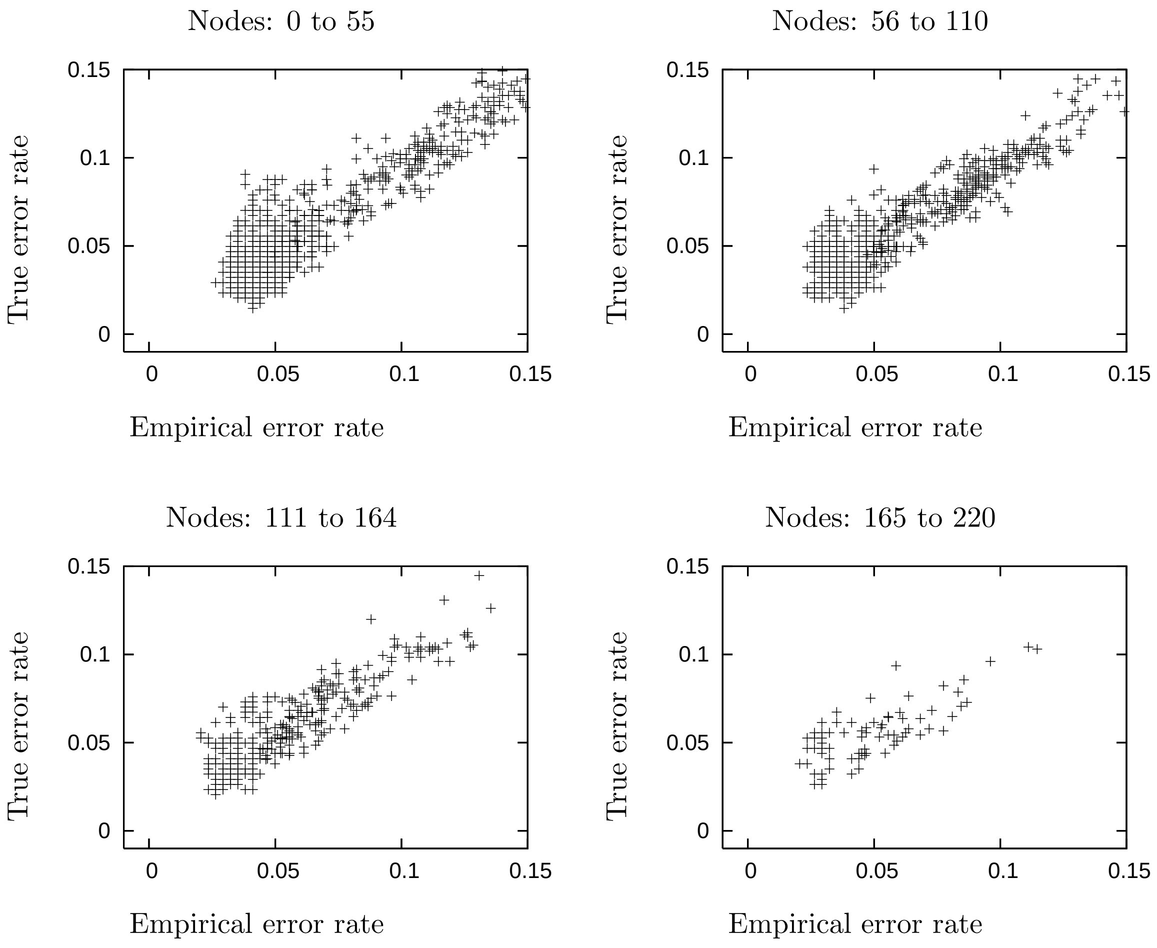


Figure 2.9: Spreads of features by GEN-ICF* with $k_{\max} = 3$ on CAR

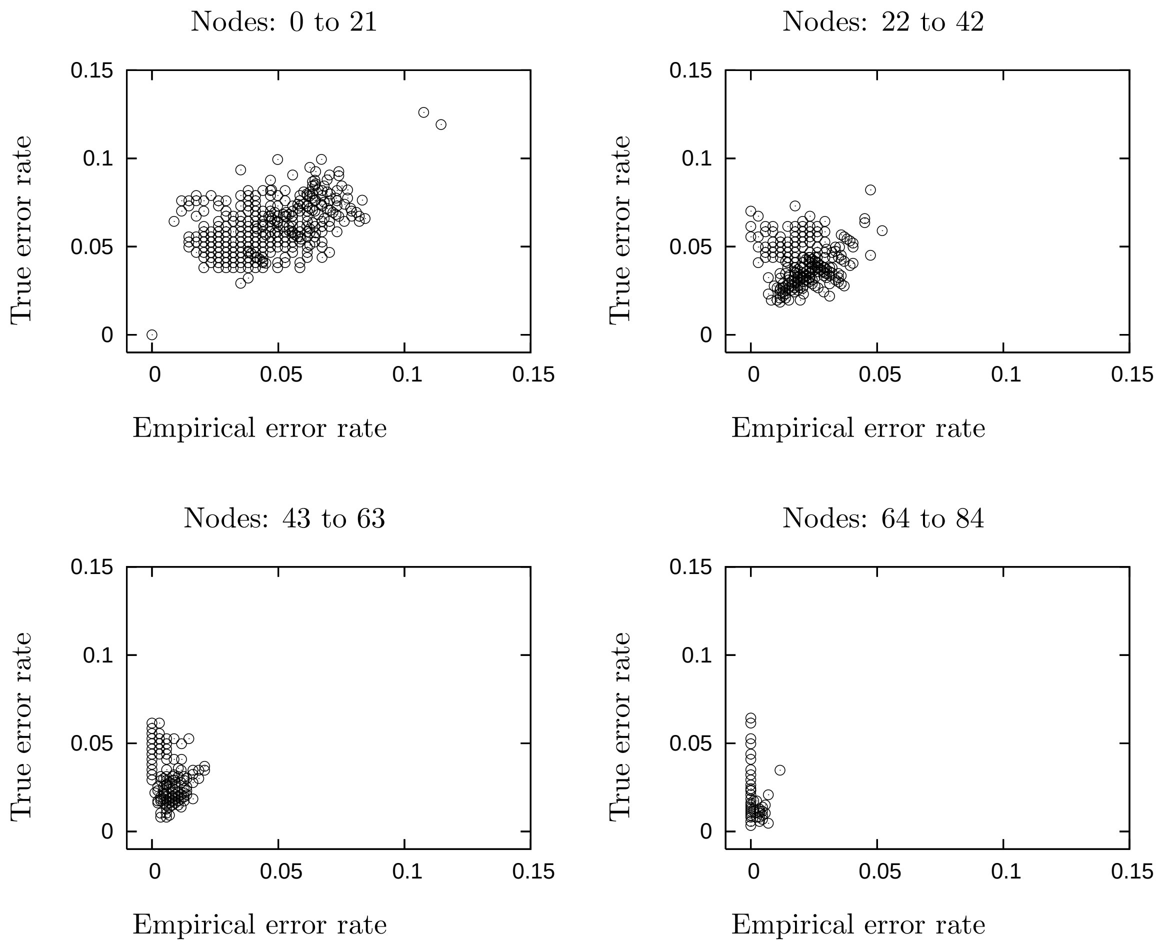


Figure 2.10: Spreads of decision trees by GEN-DT on CAR

Chapter 3

Two Construction Algorithms for ICF Classifiers: with/without Missing Bits

3.1 Introduction

In this chapter, we propose two construction algorithms for ICF classifiers: ALG-ICF and ALG-ICF*. ALG-ICF (resp., ALG-ICF*) constructs an ICF classifier by utilizing features determined by majorization (2.6). (resp., extended majorization (2.7)).

Both algorithms consist of an iteration of *composition process* and *selection process*. In the former, new features are composed from the already composed features, and in the latter, some of them are selected to be maintained for the next iteration. Then at last, an ICF classifier is obtained by selecting one feature from the maintained features as the final feature.

The two algorithms are different not only in feature types but also in selection process. ALG-ICF selects features in a greedy way on the basis of empirical error rate. On the other hand, ALG-ICF* selects features so that the selected features cover the entire training set well, as measured by the *classification cost*.

We give some computational results in Section 3.3 to make comparison between several construction algorithms; ALG-ICF, ALG-ICF* for ICF, C4.5 of release 5 [65] for DT, and BSVM of version 2.06 [48, 46, 47] for SVM. The results show that ALG-ICF* outperforms ALG-ICF, and that ALG-ICF* can construct better classifiers than C4.5 and BSVM, when its parameter values are finely tuned up. We also observe that ALG-ICF* takes much less computation time than ALG-ICF.

3.2 Algorithms ALG-ICF and ALG-ICF*

3.2.1 Common framework of ALG-ICF and ALG-ICF*

We first describe the common structure of ALG-ICF and ALG-ICF*. Both algorithms consist of two nested iterations. Let us call the h-th outer iteration ($h \geq 1$) *stage* h. An inner iteration in each stage consists of composition process and selection process.

Assume that we are in stage h. Let $F_{h'}$ denote the set of features which are generated in stage h' ($0 \leq h' < h$) and are maintained currently, where we let $F_0 = A$, the set of initial features. Let $F := F_0 \cup \cdots \cup F_{h-1}$. In the $(d-1)$-st inner iteration ($d \geq 2$) of stage h, the composition process generates the feature set $F_{h,d}$ of features as follows;

$$F_{h,d} = \{f_S \mid S \subseteq F, S \cap F_{h-1} \neq \emptyset, |S| = d\}. \tag{3.1}$$

In other words, features $\{f_S\}$ are composed in the order of the size $|S| = d$ (and thus in the order of representation complexity, approximately) in inner iterations.

The selection process then selects some features from $F \cup F_{h,2} \cup \ldots F_{h,d}$ by the *selection rule*, and prunes away the unselected features. The sets $F, F_{h,2}, \ldots, F_{h,d}$ are then reduced to the sets of such features that remain after the selection process. If the reduced set $F_{h,d}$ satisfies $F_{h,d} \neq \emptyset$, then the algorithms go to the next d-th inner iteration (and compose the set $F_{h,d+1}$ of features). On the other hand, if $F_{h,d} = \emptyset$ holds, we consider the following two cases: (i) $d = 2$ (i.e., no feature in stage h is selected). The algorithms output a feature in the current set F as an ICF classifier and halt. (ii) $d > 2$. Let us denote by $F_h = F_{h,2} \cup \cdots \cup F_{h,d-1}$. The algorithms update $F := F \cup F_h$, and go to the next stage $(h+1)$.

The common description of the two algorithms is given as follows.

Common Description of ALG-ICF and ALG-ICF*

Input: A data set X with n attributes (and thus with the set $A = \{a_1, a_2, \ldots, a_n\}$ of n initial features) and parameters (which are specified in each algorithm).

Output: An ICF classifier.

Step 1: $F := F_0 := A$ and $h := 1$.

Step 2: $d := 2$.

Step 2-1 (Composition) : Generate $F_{h,d}$ by (3.1).

Step 2-2 (Selection) : Let F' denote the features which are selected from $F \cup F_{h,2} \cup \cdots \cup F_{h,d}$ by the selection rule (specified in each algorithm). Update $F := F \cap F'$, $F_{h,2} := F_{h,2} \cap F'$, $\ldots$, and $F_{h,d} := F_{h,d} \cap F'$.

Step 2-3: If $F_{h,d} \neq \emptyset$, then let $d := d + 1$ and return to Step 2-1.

Step 3: If $d > 2$, then let $F_h := F_{h,2} \cup \cdots \cup F_{h,d-1}$, $F := F \cup F_h$, $h := h + 1$, and return to Step 2.

Step 4: Output the feature having the smallest empirical error rate among F and halt.

ALG-ICF and ALG-ICF* are different in feature types and in selection rules. In the next subsections, we show the detail of each algorithm.

3.2.2 Algorithm ALG-ICF

In ALG-ICF, each feature $f_S \in F_{h,d}$ is treated as a Boolean function $f_S : \mathbb{B}^S \rightarrow \mathbb{B}$, and is determined by (2.6).

Let $S = \{f^1, f^2, \ldots, f^d\}$ denote a set of d features, and let $S_j = S \setminus \{f^j\}$ $(j = 1, 2, \ldots, d)$. In the $(d-1)$-st inner iteration of stage h $(d \geq 2,\ h \geq 1)$, the selection rule of ALG-ICF is described as follows;

Selection Rule (ALG-ICF): A set F' of features is selected from the given sets $F \cup F_{h,2} \cup \cdots \cup F_{h,d}$ as follows.

Step 1: $F' := A$.

Step 2: For each feature $f_S \in (F \setminus A) \cup F_{h,2} \cup \cdots \cup F_{h,d}$, if f_S satisfies all the following three conditions, then $F' := F' \cup \{f_S\}$.

(i) $f_{S_j} \in F \cup F_{h,2} \cup \cdots \cup F_{h,d}$ holds for *all* $j = 1, \ldots, |S|$.

(ii) $e(f_S, X) \leq \eta e(f_{S_j}, X)$ holds for *all* $j = 1, \ldots, |S|$, where η is a parameter $(0 \leq \eta < 1)$.

(iii) f_S has the smallest empirical error rate among the features in $F \cup F_{h,2} \cup \cdots \cup F_{h,d}$ having the same $A \cap V_{f_S}$.

Proposition 2.4 tells us that empirical error rate is non-increasing as the composition proceeds. By (ii) of Step 2, we put a restrictive condition for a feature f_S to be selected.

The conditions (i) and (ii) say that a feature f_S of large S can be selected under a very tight condition, and thus f_S of a large representation complexity should not be selected. Due to the condition (iii), the number of maintained features is kept within 2^n (i.e., $|F| \leq 2^n$) during the execution of ALG-ICF.

3.2.3 Algorithm ALG-ICF*

Classification cost.

We introduce the following cost function φ to be used in the algorithm ALG-ICF*, in order to evaluate an extended feature f_S^* by the performance on the training set X.

$$\varphi(f_S^*, X) = (e(f_S^*, X) + \mu u(f_S^*, X))(\frac{1}{\Delta(f_S^*)})^{\beta}, \tag{3.2}$$

where the empirical error rate $e(f_S^*, X)$ is defined by (1.1), and

$$u(f_S^*, X) = \frac{1}{|X|}|\{x \in X \mid f_S^*(x) = *\}|,$$
$$\Delta(f_S^*) = \frac{1}{2^{|S|}}|\{s \in \mathbb{B}^S \mid f_S^*(s) \in \mathbb{B}\}|.$$

Here $u(f_S, X)$ denotes the *uncertainty rate* of f_S^* on X, and $\Delta(f_S)$ denotes the *decisiveness rate* as a function on the restricted domain $\mathbb{B}^S$ (whose size is $2^{|S|}$). μ and β are parameters to be set by the user. In the computational experiments in Section 3.3, μ is set from 0.1 to 0.5, and β is set to 0.3.

It may appear that a definition $\varphi = e + \mu u$ (i.e., (3.2) with $\beta = 0$) is more natural for the cost function; in fact, it is used in the pattern recognition algorithms [25, 67]. However, it tends to give a good score to such a feature having a high u (i.e., close to 1) if μ is small, even if the feature does not classify most examples decisively; e.g., if $e = 0$ and $u = 1$, then $\varphi = e + \mu u = \mu$. To avoid this, we require that f_S^* should be decisive to some extent, at least in such inputs s whose components are all decisive (i.e., $s \in \mathbb{B}^S$). Based on this observation, we weight the cost $e + \mu u$ by $(1/\Delta)^{\beta}$ with an appropriate parameter $\beta \geq 0$.

Given two sets $S \subset S^+$, we note that $s^+ \in \mathbb{M}^{S^+}$ and $s = s^+|_S$ satisfy $|X_{S,s}| \geq |X_{S^+,s^+}|$, since $|X_{S,s}| = \sum_{s^+|_S=s} |X_{S^+,s^+}|$. Thus as a result of introducing the term $(1/\Delta)^{\beta}$, we expect that $\varphi(f_S^*, X) \leq \varphi(f_{S^+}^*, X)$ holds, since $f_{S^+}^*(s^+) = *$ may hold for a small $|X_{S^+,s^+}|$ under a relatively small α (e.g., $\alpha \leq 0.5$), as indicated in Figure 2.2; it leads to a small $\Delta(f_{S^+}^*)$ and thus a large $\varphi(f_{S^+}^*, X)$. In summary, the cost function (3.2) with a reasonably large β gives an advantage to such a feature f_S^* that attains a small $e + \mu u$ and is composed of a small set S. In this sense, features selected by the selection process of ALG-ICF* are kept small representation complexity, and thus kept rather *robust* (i.e., not overfitting to the data set X). We will discuss the influence of parameters α, β, μ later in Section 3.3.

Description of ALG-ICF*.

In ALG-ICF*, each feature $f_S^* \in F_{h,d}$ is treated as $f_S^* : \mathbb{M}^S \to \mathbb{M}$ determined by (2.7).

Consider the $(d-1)$-st inner iteration of stage h ($d \geq 2$, $h \geq 1$). For each example $x \in X$, we define

$$F(x) = \{f_S^* \in (F \setminus A) \cup F_{h,2} \cup \cdots \cup F_{h,d} \mid f_S^*(x) = y(x)\}, \tag{3.3}$$

i.e., $F(x)$ is the set of features (not including initial features) covering x. The selection process of ALG-ICF* tries to maintain a set of features from $F \cup F_{h,2} \cup \cdots \cup F_{h,d}$ so that the resulting features, as partial knowledge, cover X well. To be more precise, for each example $x \in X$, if $F(x) \neq \emptyset$, the feature $f_S^* \in F(x)$ which has the smallest $\varphi(f_S^*, X)$ is selected; features not selected for any example $x \in X$ are pruned away.

Also, among the features selected as above, ones having relatively large φ are also pruned. The selection process of ALG-ICF* is described as follows.

Selection Rule (ALG-ICF*): A set F' of features is selected from the given sets $F \cup F_{h,2} \cup \cdots \cup F_{h,d}$ as follows.

Step 1: $F' := \emptyset$. If $d > 2$, for each $f_S^* \in F_{h,d}$, test if there is an $f_{S_j}^* \in F_{h,d-1}$ for *some* $j = 1, \ldots, d$. If no, then let $F_{h,d} := F_{h,d} \setminus \{f_S^*\}$.

Step 2: Using the resulting $F, F_{h,2}, \ldots, F_{h,d}$, construct $F(x)$ of (3.3) for all $x \in X$. For each $x \in X$, if $F(x) \neq \emptyset$, then choose $f_S^* \in F(x)$ having the smallest $\varphi(f_S^*, X)$. If $f_S^* \notin F'$, then let $F' := F' \cup \{f_S^*\}$.

Step 3: Exclude $\lfloor \pi |F'| \rfloor$ features from F' having the largest φ, where π is a parameter ($0 \leq \pi \leq 1$).

Step 4: $F' := F' \cup A$.

Note that initial features are not pruned by the selection process. It is due to the empirical reason that maintaining them often makes a constructed classifier better.

Since at most one feature is selected for one example $x \in X$, $|F| \leq n + |X|$ holds at the end of the selection process. This bound is much smaller than the bound $|F| \leq 2^n$ for ALG-ICF.

ALG-ICF* aims at constructing *global knowledge* (i.e., the output classifier) from pieces of partial knowledge. We expect the features generated in later stages to attain small e and u, $\Delta \simeq 1$, and thus a small φ. Here, boosting (e.g., [35]) is a methodology to construct a better classifier from a set of classifiers, called *weak hypotheses*. In the research of boosting, it is pointed out that weak hypotheses covering different examples from each other result in a good classifier. We expect that our selection rule with covering condition has a similar effect.

In Step 4 of the common algorithm description of Section 3.2.1, we modify each feature $f_S^* \in F$ to a function $f_S : \mathbb{M}^S \to \mathbb{B}$ by determining $f_S(s) = 0$ or 1 by (2.6) for each $s \in \mathbb{M}^S$. Then the empirical error rate $e(f_S, X)$ is computed again. We obtain an ICF classifier as the feature attaining the smallest empirical error rate.

3.3 Computational experiments

In the experiments, we compare classifiers constructed by four construction algorithms based on different representation models. All computations are carried out on a PC (Pentium IV 2.8GHz, memory 1GB).

3.3.1 Experimental setting

Data sets.

We utilize UCI data sets of Table 2.1 by transforming them into binary data sets, similarly to the experiments in Chapter 2. We also utilize *artificial data sets*. Each artificial data set has a linear threshold function as its oracle $y : \mathbb{B}^n \to \mathbb{B}$, which is defined as follows: For each $x \in \mathbb{B}^n$,

$$y(x) = \begin{cases} 1 & \text{if } \sum_{j=1}^{n} w_j x_j \geq 0, \\ 0 & \text{otherwise,} \end{cases}$$

where we use $n = 14$. Let $n_0 \in \{5, 6, \ldots, n\}$. For an attribute $j = 1, 2, \ldots, n_0$, we set w_j at random so that $-1 \leq w_j \leq 1$, and for a remaining attribute $j = n_0 + 1, n_0 + 2, \ldots, n$, we set $w_j = 0$, in order to allow irrelevant attributes. (Thus each value of n_0 defines an oracle on $\mathbb{B}^n$.) For each $n_0 \in \{5, 6, \ldots, n\}$, we call the data set containing whole vectors in $\mathbb{B}^n$ as the examples ART5, ART6, ..., ART14, respectively.

For each data set X and each construction algorithm, we generate a training set X_{train} and a test set X_{test} 10 times. For a UCI data set, we generate X_{train} and X_{test} by dividing X into halves at random so that $|X_{\text{train}}| = |X_{\text{test}}| = |X|/2$ and $X_{\text{train}} \cap X_{\text{test}} = \emptyset$ hold. For an artificial data set, we pick up 400 vectors in $\mathbb{B}^n$ at random and use them as examples of X_{train}. We use the whole vectors in $\mathbb{B}^n$ as the test set, i.e,. $X_{\text{test}} = \mathbb{B}^n$. We construct a classifier from each X_{train}, and estimate the true error rate by the average of 10 error rates of classifiers on their test sets.

Parameter values.

The construction algorithms have some program parameters, and they have more or less significant influence on the performance.

ALG-ICF has a single parameter $\eta \in [0, 1)$, which controls the number of generated features; if η is larger, then more features are generated and the computation time gets larger. We use $\eta \in \{0.75, 0.80, 0.85, 0.90, 0.95\}$ such that ALG-ICF halts within 600 seconds. Also, we use $b = 0$ in majorization (2.6).

For ALG-ICF*, we use all combinations of parameter values such that $\alpha \in \{0.01, 0.05, 0.1, 0.25, 0.5\}$, $\beta = 0.3$, $\mu \in \{0.1, \ldots, 0.5\}$, and $\pi = 0.3$. The influence of α, β, μ on ALG-ICF* is discussed in the next subsection.

Among the parameters of C4.5 (see Section 2.3.2), CF (confidence level) has the most significant role in determining the performance of output decision trees. We use 1%, 5%, 10%, 25%, 50%, 75%, 100% for CF.

A classifier of SVM is represented in the form of hyperplane, and its construction is formulated in several ways [54, 72, 38, 78]. BSVM [48, 46, 47] uses a standard formulation called a *generalized support vector machine* in [54]. In this formulation, an SVM classifier r is represented as follows;

$$r(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^{m} w_i y'_i K(x, x^i) \geq \theta, \\ 0 & \text{otherwise,} \end{cases} \tag{3.4}$$

where $m = |X_{\text{train}}|$, $x^i \in X_{\text{train}}$ is the i-th example in X_{train}, $y'_i = 2y(x^i) - 1$ (and thus $y'_i \in \{\pm 1\}$), $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a *kernel function*, each w_i is determined by solving the following mathematical programming problem;

$$\begin{aligned} \max_{w_1, w_2, \ldots, w_m} \quad & \sum_{i=1}^{m} w_i - \frac{1}{2} \sum_{i,\iota=1}^{m} w_i w_k y'_i y'_\iota K(x^i, x^\iota) \\ \text{s.t.} \quad & \sum_{i=1}^{m} w_i y'_i = 0, \\ & w_i \geq 0 \quad (i = 1, 2, \ldots, m), \end{aligned} \tag{3.5}$$

and θ in (3.4) is determined from the obtained $w_1, w_2, \ldots, w_m$.

For the kernel function, we employ two types of Boolean kernel functions for binary data sets, denoted by K_2 and K_3, which are defined by:

$$\begin{aligned} K_2(x, x^i) &= 2^{|\{j \mid x_j = x^i_j = 1,\ j = 1, 2, \ldots, n\}|} - 1, \\ K_3(x, x^i) &= 3^{|\{j \mid x_j = x^i_j,\ j = 1, 2, \ldots, n\}|} - 1. \end{aligned}$$

Table 3.1: Best true error rates ($\times 10^2$) of construction algorithms on UCI data sets

Data	ALG-ICF	ALG-ICF*	C4.5	BSVM	(Std.err)
AUS	**15.13**	$\star$**14.87**	**15.13**	**16.03**	1.32
BCW	**4.33**	**4.09**	5.15	$\star$**3.92**	0.74
BUPA	37.05	**33.35**	35.85	$\star$**33.29**	2.54
CAR	5.53	1.10	1.81	$\star$**0.57**	0.19
CRX	$\star$**12.91**	**13.43**	**13.69**	15.57	1.31
FLAG	**10.62**	**10.62**	$\star$**10.60**	13.20	2.21
HABER	**27.62**	$\star$**26.60**	**26.95**	31.98	2.58
HEART	22.15	$\star$**18.30**	**20.15**	21.33	2.35
IONO	$\star$**10.00**	11.70	**10.29**	12.27	1.60
MUSH	1.14	$\star$**0.00**	$\star$**0.00**	$\star$**0.00**	0.00
PIMA	26.38	$\star$**24.69**	**26.04**	28.83	1.56
TTT	19.71	4.05	13.13	$\star$**1.71**	0.42
VOTES	$\star$**3.99**	**4.13**	**4.86**	5.41	0.94
Best	15.12	$\star$12.84	14.13	14.16	(N.A.)

Note that the original implementation of BSVM does not support Boolean kernel functions. We implement these functions by ourselves.

3.3.2 Results

True error rates.

We examine the best (i.e., smallest) true error rates of each construction algorithm, obtained from all possible parameter values. The results for UCI data sets and artificial data sets are shown in Tables 3.1 and 3.2, respectively. In the table, the sign $\star$ indicates the best true error rate e among the four construction algorithms, and its standard error represented in the rightmost column is determined by $\sqrt{e(1-e)/|X|}$. We represent the error rates whose difference from e is within the standard error (i.e., no significant difference from e) by boldface. Also, at the bottom of Table 3.1 (resp., Table 3.2), we show the average of error rates for all UCI data sets (resp., artificial data sets).

We see that ALG-ICF* is the best construction algorithm among all in the sense that it ranks first on most data sets, and even in the data sets where it is not the case, it ranks second or its error rates are within standard errors. Also, BSVM may be most suitable to artificial data sets since an SVM classifier is represented by hyperplane, but for small n_0, ALG-ICF*

Table 3.2: Best true error rates ($\times 10^2$) of construction algorithms on artificial data sets

Data	ALG-ICF	ALG-ICF*	C4.5	BSVM	(Std.err)
ART5	15.63	⋆**0.00**	⋆**0.00**	1.76	0.00
ART6	15.78	⋆**0.00**	0.63	3.78	0.00
ART7	19.30	⋆**0.86**	3.03	5.41	0.07
ART8	14.80	⋆**2.50**	4.72	5.88	0.12
ART9	17.48	⋆**3.53**	6.55	5.82	0.14
ART10	17.54	⋆**5.47**	10.00	7.58	0.18
ART11	19.79	⋆**6.92**	10.67	7.80	0.20
ART12	23.02	**8.55**	13.95	⋆**8.52**	0.22
ART13	28.81	10.99	17.98	⋆**9.04**	0.22
ART14	23.09	12.26	19.64	⋆**9.18**	0.23
BEST	19.52	⋆5.11	8.72	6.48	(N.A.)

attains the best performance among all. The results show that ALG-ICF* *can* construct a good classifier for various kinds of data sets by using appropriate parameter values.

In Figure 3.1, we show an ICF classifier constructed by ALG-ICF* on BCW data set: For each feature f_S^*, we put the name of the feature f_S^*, the names of the features in S, and the vectors $s \in \mathbb{M}^S$ to which $f_S^*(s) = 0$ or 1 holds (those to $f_S^*(s) = *$ is omitted). Also, the estimated true error rate is shown on the final node (i.e., it is 4.971%, present on the node named `5-2`).

Computation time.

Table 3.3 shows the average of computation time which each construction algorithm takes to construct the classifiers given in Tables 3.1 and 3.2. In the table, the sign - indicates that computation time is less than 0.01 seconds. For ALG-ICF, recall that we test several values for the parameter η. Then the sign + indicates that ALG-ICF does not halt within 600 seconds for large values of η, which means that the error rates given in Tables 3.1 and 3.2 are obtained by a small value of η.

As can be seen, C4.5 outperforms the other algorithms. BSVM ranks second, and ALG-ICF* is slightly worse than BSVM. We note that ALG-ICF* takes much less computation time than ALG-ICF. In our experience, the computation time of ALG-ICF* and ALG-ICF is proportional to $|F|$, the number of maintained features, which is bounded as $|F| \leq n + |X|$ in ALG-ICF* and $|F| \leq 2^n$ in ALG-ICF. It may explain the significant difference in the computation time between the two construction algorithms.

Comparison of ALG-ICF and ALG-ICF*.

As clear from the computational results, ALG-ICF* shows much better performance than ALG-ICF both in true error rate and in computation time. Recall that ALG-ICF* and ALG-ICF are different in (i) feature types and in (ii) selection rules.

The effect of (i) can be seen in our observation that, for almost all data sets, the best ALG-ICF* classifiers have small depths in DAG representations, and are achieved by small α from 0.01 to 0.1 (i.e., composite features output $*$ for a nontrivial portion of inputs).

The effect of (ii) is also significant. ALG-ICF* maintains not only major features covering many examples but also minor features covering a small number of (exceptional) examples in the data set; ALG-ICF may maintain only major features due to its selection rule. We also conducted the computational comparison between ALG-ICF* and ALG-ICF with features $f_S^* : \mathbb{M}^S \to \mathbb{M}$ determined by extended majorization (2.7) (i.e., only the selection rules are different), and observed that the former still gives better true error rate than the latter (the details are omitted).

Influence of parameters on ALG-ICF*.

We discuss the influence of parameters α, β, μ on ALG-ICF*. Let S and S^+ denote arbitrary sets of features such that $S \subset S^+$. Take an arbitrary vector $s^+ \in \mathbb{B}^{S^+}$, and let $s = s^+|_S$.

We first consider the influence of α. If α is large (e.g., $\alpha \geq 0.75$), then $f_S^*(s)$ and $f_{S^+}^*(s^+)$ are likely to be set to 0 or 1, and $\Delta(f_S^*)$ and $\Delta(f_{S^+}^*)$ become close to 1 (hence so do $(1/\Delta(f_S^*))^\beta$ and $(1/\Delta(f_{S^+}^*))^\beta$ in (3.2)). Then the influence of Δ, u on $\varphi(f_S^*, X), \varphi(f_{S^+}^*, X)$ should be small, and Proposition 2.4 tells us that $\varphi(f_{S^+}^*) \leq \varphi(f_S^*)$ (approximately) holds. $f_{S^+}^*$ composed of a large set S^+ is preferred in the selection process, and thus the inner iteration tends to halt with a large d.

On the other hand, if α is small, then since $|X_{S^+,s^+}| < |X_{S,s}|$, $f_{S^+}^*(s^+)$ is likely to be set to $*$, whereas $f_S^*(s)$ is still likely to be set to 0 or 1 (see Figure 2.2). In this case, $\Delta(f_{S^+}^*) < \Delta(f_S^*)$ implies $(1/\Delta(f_{S^+}^*))^\beta > (1/\Delta(f_S^*))^\beta$ (unless β is extremely small). Then f_S^* composed of a small set S is preferred in the selection process, and thus the inner iteration tends to halt with a small d.

We next consider the influence of β and μ. As argued above, $u(f_S^*, X) \leq u(f_{S^+}^*, X)$ and $\Delta(f_{S^+}^*) \leq \Delta(f_S^*)$ should usually hold (and $\mu u(f_S^*, X) \leq \mu u(f_{S^+}^*, X)$ and $(1/\Delta(f_{S^+}^*))^\beta \geq (1/\Delta(f_S^*))^\beta$ should also hold). Then, if β and μ are large, we have $\varphi(f_{S^+}^*, X) > \varphi(f_S^*, X)$, and f_S^* composed of a small set S is preferred in the selection process, and thus the inner iteration tends to halt with a small d. If β and μ are small, the converse would be observed.

Figure 3.2 illustrates the above discussion about the influence of α, β on the size of d in one stage of ALG-ICF*, where a heavier color indicates a large d. Figure 3.3 gives the results observed on ART8 data set. As the inner iteration may be executed more than once (i.e., in more than one stage) in one execution of ALG-ICF*, we keep the maximum $d_{\max}$ of d among all stages, and take the average over 10 training sets for given α, β, μ. In Figure 3.3, we show the average of $d_{\max}$ by colors for each $\mu \in \{0.2, 0.3, 0.4, 0.5\}$, where a heavier color means a large $d_{\max}$. We notice that the results show a tendency similar to that anticipated in Figure 3.2.

Note that, if d is too large (resp., too small), then the resulting features may overfit (resp., underfit) to the data set; in either case, they may attain poor true error rates. Thus we should determine α, β, μ so that d is "appropriate" for the considered data set.

To confirm the above observation, we consider true error rates realized by different values of α, β, μ (leading to different d) on ART8 data set. For given α, β, μ, we denote the estimated true error rate by $\varepsilon(\alpha, \beta, \mu)$. Figure 3.4 gives $\varepsilon(\alpha, \beta, \mu)$ on ART8 for all tested α, β, μ. The tendency as discussed above is clearly shown here, since the middle areas attain rather small true error rates, where $d_{\max}$ is from 5 to 7 approximately.

The results in Tables 3.1 and 3.2 tell that ALG-ICF* *can* construct good classifiers if we are allowed to tune the parameters appropriately, which, however, may be difficult to be attained in practical situations.

3.4 Concluding Remarks

We proposed two construction algorithms for the representation model ICF, ALG-ICF and ALG-ICF*. ALG-ICF is based on features determined by majorization (2.6), while ALG-ICF* is based on those determined by extended majorization (2.7). Both algorithms consist of composition process and selection process, but are significantly different in selection process. Our computational experiments show that ALG-ICF* outperforms ALG-ICF both in true error rate and in computation time, and that ALG-ICF* is better than C4.5 for DT and BSVM for SVM in true error rate.

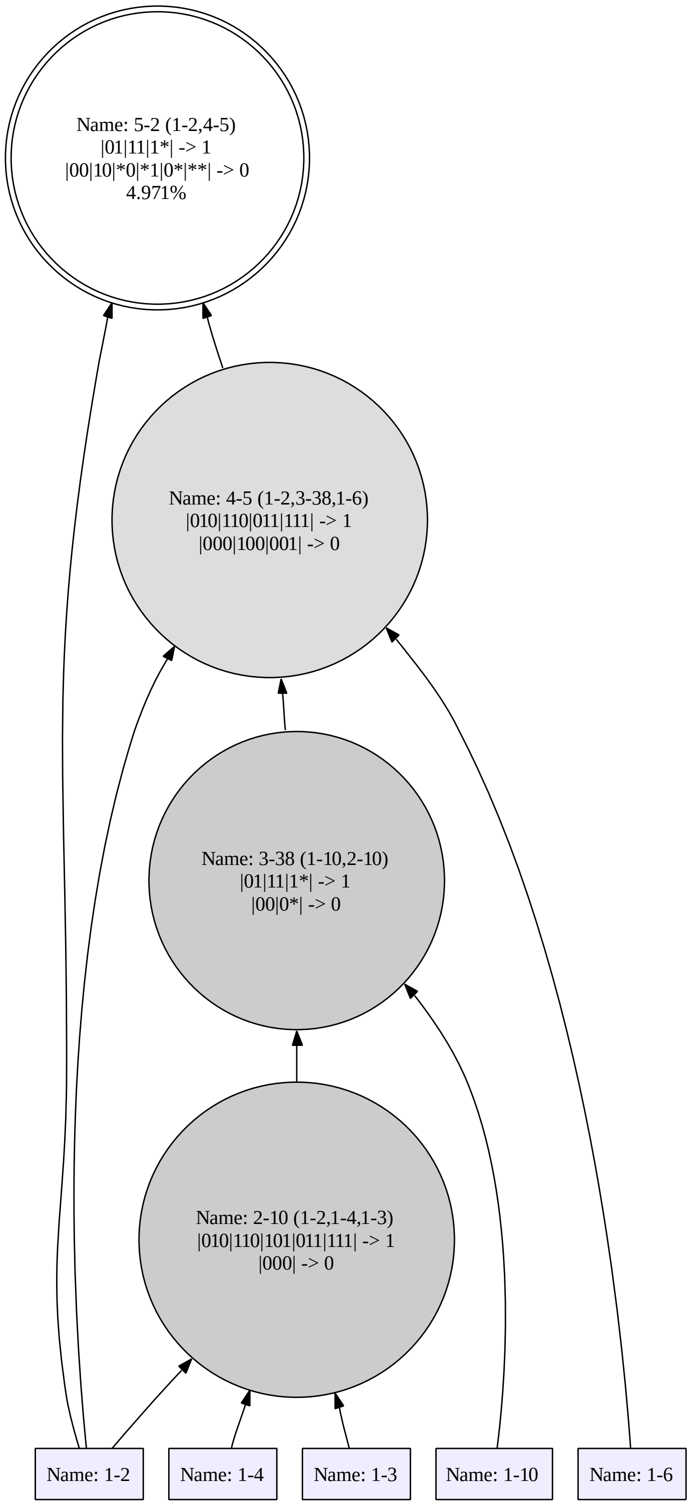


Figure 3.1: An ICF classifier composed on BCW

Table 3.3: Computation time (sec.) of construction algorithms

Data	ALG-ICF	ALG-ICF*	C4.5	BSVM
AUS	+0.03	0.11	-	0.06
BCW	+15.48	0.51	-	0.02
BUPA	+0.16	1.73	-	0.02
CAR	+31.34	5.71	-	0.11
CRX	+0.02	0.11	-	0.05
FLAG	-	-	-	-
HABER	+2.45	0.08	-	-
HEART	+0.05	0.06	-	0.01
IONO	0.01	0.02	-	0.01
MUSH	0.29	0.24	0.01	0.10
PIMA	+1.50	0.14	0.01	0.08
TTT	+26.59	9.07	-	0.09
VOTES	0.02	0.02	-	0.01
ART5	+0.36	0.08	-	0.04
ART6	+0.57	0.18	-	0.04
ART7	+0.86	0.55	-	0.04
ART8	+5.92	0.39	-	0.04
ART9	+0.88	0.52	-	0.04
ART10	+4.83	0.73	-	0.04
ART11	+1.17	0.47	-	0.04
ART12	+0.66	0.69	-	0.04
ART13	+0.21	2.34	-	0.05
ART14	+11.70	1.43	-	0.05

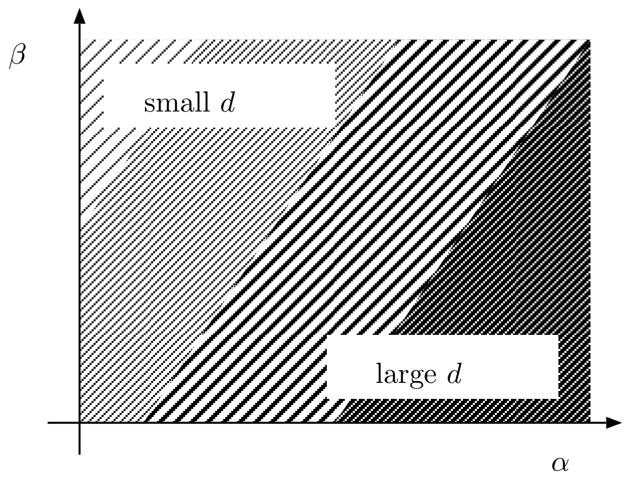


Figure 3.2: Influence of α, β on the resulting d of inner iterations

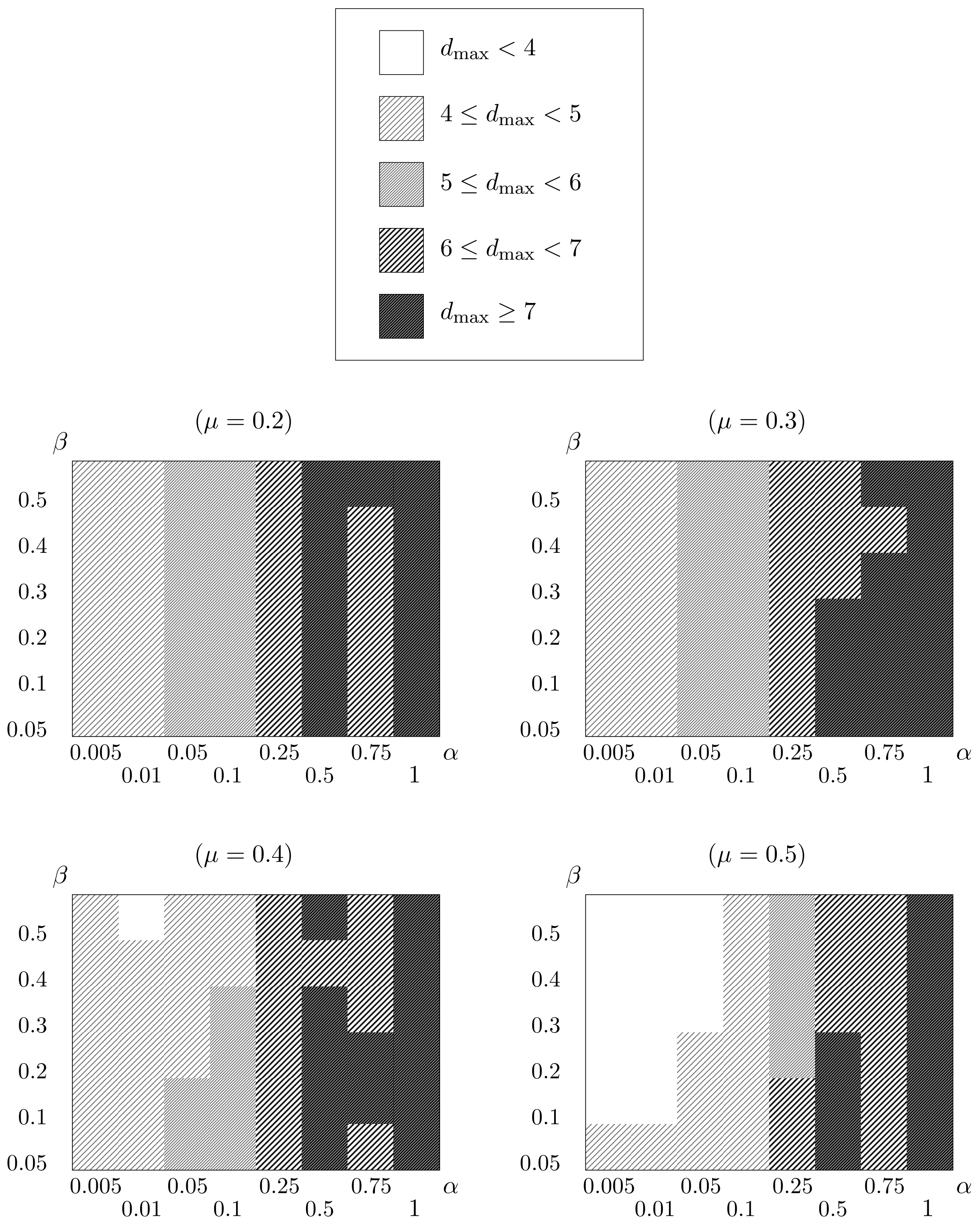


Figure 3.3: $d_{\max}$ for various α, β, μ on ART8

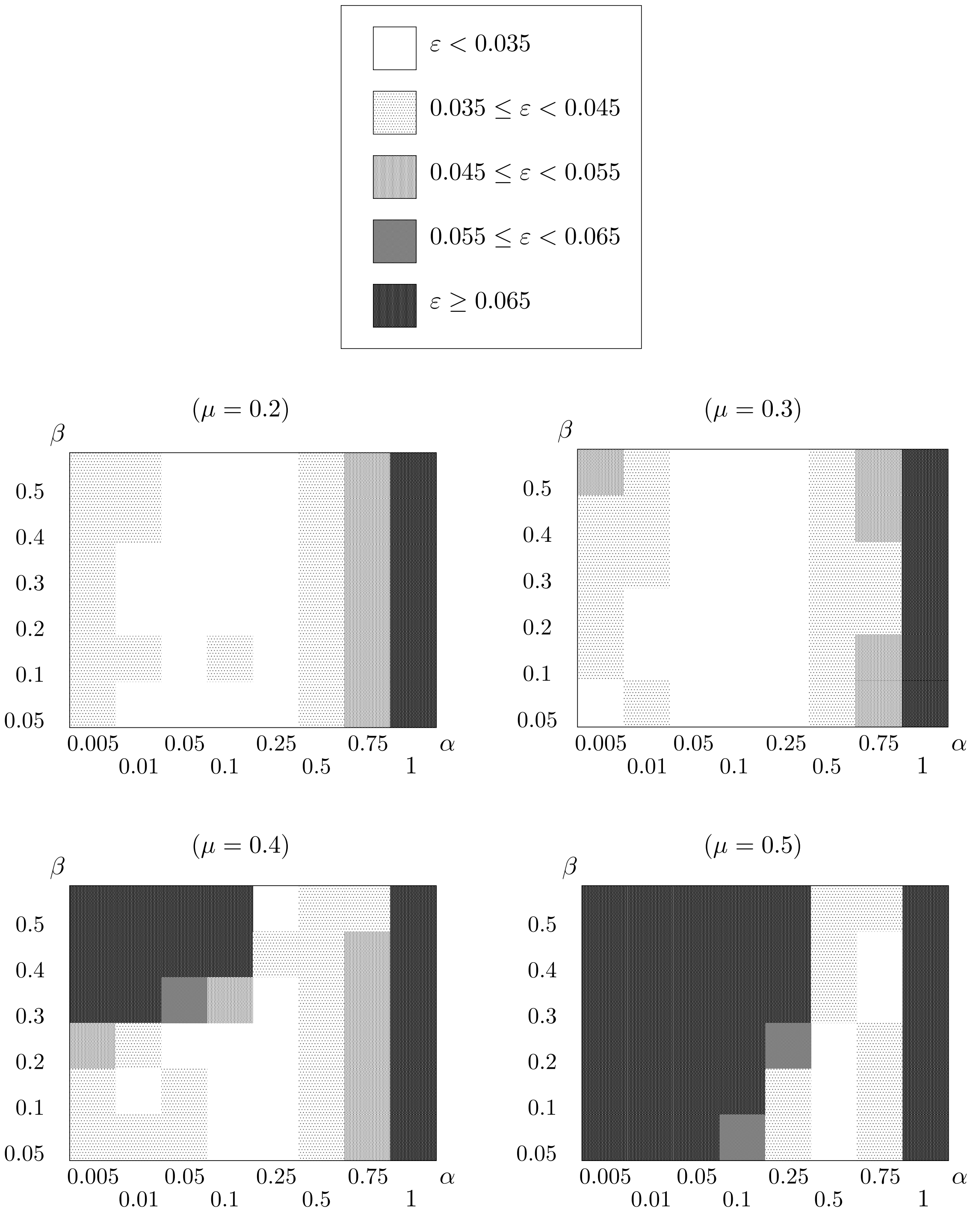


Figure 3.4: $\varepsilon(\alpha, \beta, \mu)$ on ART8

Chapter 4

Extention of ICF Classifiers to Real World Data Sets

4.1 Introduction

In Chapter 3, we proposed the construction algorithm ALG-ICF* of ICF classifiers on $\mathbb{M}$-valued data sets, i.e., the data space $\mathbb{S}$ is represented as $\mathbb{S} = \mathbb{M}^n$, where n denotes the dimensionality of $\mathbb{M}$-valued data sets. In this chapter, we extend ALG-ICF* so that it can process real world data sets consisting of numerical and/or categorical attributes. For this purpose, we incorporate a discretization scheme into ALG-ICF* as its preprocessor, by which an input real world data set is transformed into $\mathbb{M}$-valued one.

Let us denote by N the dimensionality of such a real world data set. For each attribute $q \in \{1, 2, \ldots, N\}$, let us denote by $\mathbb{D}_q$ the *domain* of the attribute q. We assume that $\mathbb{D}_q$ is either numerical or categorical. For a numerical attribute q, $\mathbb{D}_q$ may be a subset of the set $\mathbb{R}$ of real numbers or the set $\mathbb{Z}$ of integers. For a categorical attribute q', $\mathbb{D}_{q'}$ is a set of unordered categories, e.g., $\mathbb{D}_{q'} = \{\texttt{blue}, \texttt{red}, \texttt{white}, \texttt{black}\}$. We assume the data space $\mathbb{S}$ to be $\mathbb{S} = \mathbb{D}_1 \times \mathbb{D}_2 \times \cdots \times \mathbb{D}_N$, and write a data set over $\mathbb{S}$ by Ω.

In order to process Ω by ALG-ICF*, we equip ALG-ICF* with a discretization scheme as its processor, which maps Ω over $\mathbb{S}$ to X over $\mathbb{M}^n$ (where the dimensionality n is suitably determined by the discretization scheme): In the resulting algorithm, we first transform Ω into a data set X and then apply ALG-ICF* to X in order to construct an ICF classifier. Thus our purpose in this chapter is to establish such a discretization scheme.

Let us denote by χ a *discretizer*, which is a mapping from $\mathbb{S}$ to $\mathbb{M}$. Our discretization scheme constructs a set $D = \{\chi_1, \chi_2, \ldots, \chi_n\}$ of n discretizers, where each χ_j $(j = 1, 2, \ldots, n)$ will be used as attribute j in the transformed $\mathbb{M}$-valued data set X. It is desirable for a discretization scheme to select discretizers so that we can construct a good ICF classifier by ALG-ICF* from the resulting X.

In Section 4.2, we introduce two conventional discretization schemes, domain based construction (DC) and space based construction (SC). In DC, we regard a discretizer as a classifier with reject option (i.e., classifier which may output $*$ to indicate "we don't know the class" rather than 0 or 1). For each numerical or categorical attribute $q \in \{1, 2, \ldots, N\}$, we select such a discretizer that minimizes the *misclassification cost* among all candidates. We can find such an optimum discretizer by solving dynamic programming for a numerical attribute [37], and by the Naïve-Bayesian approach for a categorical attribute [28]. On the other hand, the SC searches a set of discretizers that partitions the data space $\mathbb{S}$ into "well-separated" subspaces by a greedy algorithm.

Then in Section 4.3, after studying the advantages and defects of these schemes by computational experiments, we propose algorithm ALG-ICF$^*_{\mathrm{IC}}$, which is ALG-ICF* equipped with a new discretization scheme, integrated construction (IC). Our computational experiments reveal that ALG-ICF$^*_{\mathrm{IC}}$ outperforms C4.5 designed for real world data sets in many cases.

4.2 Conventional discretization schemes

4.2.1 Discretizers

For a numerical or categorical attribute $q \in \{1, 2, \ldots, N\}$, we define a discretizer by a tuple $\chi = (q, \mathcal{P}, \ell)$, where $\mathcal{P}$ denotes a *partition* of the domain $\mathbb{D}_q$ and ℓ denotes a *label*. A partition $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ is a family of disjoint subsets of $\mathbb{D}_q$, i.e.,

$$\bigcup_{\kappa=1,2,\ldots,k} P_\kappa = \mathbb{D}_q, \qquad\qquad P_\kappa \cap P_{\kappa'} = \emptyset \quad (1 \le \kappa < \kappa' \le k). \tag{4.1}$$

For a numerical attribute q, we assume that $\mathbb{D}_q$ is a closed interval $[\min \mathbb{D}_q, \max \mathbb{D}_q]$, and define a partition $\mathcal{P}$ by *cutpoints*: For $w_1, w_2, \ldots, w_{k-1} \in \mathbb{D}_q$, we take the k intervals $[\min \mathbb{D}_q, w_1)$, $[w_1, w_2)$, $\ldots$, $[w_{k-2}, w_{k-1})$, $[w_{k-1}, \max \mathbb{D}_q]$ as the elements of $\mathcal{P}$, respectively (where we assume $\min \mathbb{D}_q < w_1 < w_2 < \cdots < w_{k-1} \le \max \mathbb{D}_q$). For a categorical attribute, $\mathcal{P}$ is determined by a family of subsets $P_1, P_2, \ldots, P_k$ of categories satisfying (4.1).

A label ℓ is a mapping from $\{1, 2, \ldots, k\}$ to $\mathbb{M}$, i.e., ℓ assigns a value in $\mathbb{M}$ to each of partitioned subsets $P_1, P_2, \ldots, P_k$. Then a discretizer $\chi = (q, \mathcal{P}, \ell)$ discretizes a data element $\omega \in \mathbb{S}$ as follows: Let us denote by $\kappa \in \{1, 2, \ldots, k\}$ the index such that $\omega_q \in P_\kappa$ holds. Then ω is mapped to $\ell(\kappa)$.

Table 4.1: A real data set $\Omega = \Omega^1 \cup \Omega^0$

		ω_1	ω_2	ω_3
Ω^1	ω^1	10.0	`blue`	-1
	ω^2	6.3	`red`	0
Ω^0	ω^3	8.5	`white`	2
	ω^4	4.1	`white`	0
	ω^5	3.5	`black`	1

Let us take an example on a data set Ω of Table 4.1. Assume a discretizer χ_1 as follows:

$$\chi_1 = \big(1, \{[-\infty, 8.0), [8.0, +\infty]\}, \ell_1\big),$$

where ℓ_1 is a label such that $\ell(1) = 0$ and $\ell(2) = 1$. Then χ_1 maps the example $\omega^1 = (10.0, \texttt{blue}, -1)$ to $\ell(2) = 1$ since the value 10.0 of attribute 1 is in the second partitioned subset $[8.0, +\infty]$. χ_1 maps the other examples ω^2, ω^3, ω^4, and ω^5 to 0, 1, 0, and 0, respectively.

For $\chi = (q, \mathcal{P}, \ell)$ and $\omega \in \mathbb{S}$, we write the mapped value by $\chi(\omega)$ instead of $\ell(\kappa)$ for convenience (where $\omega_q \in P_\kappa$). For a set $D = \{\chi_1, \chi_2, \ldots, \chi_n\}$ of discretizers, we write $D(\omega) = (\chi_1(\omega), \chi_2(\omega), \ldots, \chi_n(\omega))$ and $D(\Omega) = \{D(\omega) \mid \omega \in \Omega\}$.

In the discretization schemes DC and SC, a discretizer $\chi = (q, \mathcal{P}, \ell)$ is determined by q and $\mathcal{P}$, which means that ℓ is determined uniquely by q and $\mathcal{P}$, but in a different way between the two schemes. How to determine ℓ and how to select q and $\mathcal{P}$ in each scheme are described in the subsequent subsections.

4.2.2 Discretization scheme DC

Based on previous discretization schemes [28, 31], this subsection shows the discretization scheme DC. It constructs a set $D = \{\chi_1, \chi_2, \ldots, \chi_n\}$ of discretizers with $n = N$, which means that one discretizer is constructed from one attribute. DC has two parameters, K and μ', where K specifies the maximum cardinality of partitions for numerical attributes, and $\mu' \in [0, 1]$ denotes the cost incurred by an assignment of $*$ to a discretizer.

How to determine a label.

Let us take a partition $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ on attribute q. By q and $\mathcal{P}$, the data set Ω is partitioned into k subsets according to the values of attribute q as $\Omega_{q,\mathcal{P},\kappa} = \{\omega \in \Omega \mid \omega_q \in P_\kappa\}$, $\kappa \in \{1, 2, \ldots, k\}$. We denote $\Omega^1_{q,\mathcal{P},\kappa} = \Omega^1 \cap \Omega_{q,\mathcal{P},\kappa}$ and $\Omega^0_{q,\mathcal{P},\kappa} = \Omega^0 \cap \Omega_{q,\mathcal{P},\kappa}$.

For given q and $\mathcal{P}$, we consider determining a value $\ell(\kappa) \in \mathbb{M}$ for $\kappa \in \{1, 2, \dots, k\}$ so that the cost defined in the following is minimized: If we assign $\ell(\kappa) = 1$ (resp., 0), then it costs us $|\Omega^0_{q,\mathcal{P},\kappa}|$ (resp., $|\Omega^1_{q,\mathcal{P},\kappa}|$) since the examples in $\Omega^0_{q,\mathcal{P},\kappa}$ (resp., $|\Omega^1_{q,\mathcal{P},\kappa}|$) are classified erroneously if we regard ℓ as a classifier with reject option. On the other hand, if we assign $\ell(\kappa) = *$, then it costs us $\mu'|\Omega_{q,\mathcal{P},\kappa}|$, where the parameter μ' is used as the relative cost of an uncertain decision to an erroneous one. Then we define the misclassification cost $\Gamma_{\mathrm{DC}}(\chi)$ of a discretizer $\chi = (q, \mathcal{P}, \ell)$ as the sum of such costs over the k partitioned subsets $\Omega_{q,\mathcal{P},1}, \Omega_{q,\mathcal{P},2}, \dots, \Omega_{q,\mathcal{P},k}$;

$$\Gamma_{\mathrm{DC}}(\chi) = \sum_{\kappa=1,2,\dots,k:\ \ell(\kappa)=1} |\Omega^0_{q,\mathcal{P},\kappa}| + \sum_{\kappa=1,2,\dots,k:\ \ell(\kappa)=0} |\Omega^1_{q,\mathcal{P},\kappa}| + \mu' \sum_{\kappa=1,2,\dots,k:\ \ell(\kappa)=*} |\Omega_{q,\mathcal{P},\kappa}|. \tag{4.2}$$

Since the cost to each $\Omega_{q,\mathcal{P},\kappa}$ is computed independently, the misclassification cost is minimized by the following label: For $\kappa \in \{1, 2, \dots, k\}$,

$$\ell(\kappa) = \begin{cases} * & \text{if } \mu'|\Omega_{q,\mathcal{P},\kappa}| \le \min\{|\Omega^1_{q,\mathcal{P},\kappa}|, |\Omega^0_{q,\mathcal{P},\kappa}|\}, \\ 1 & \text{if } \mu'|\Omega_{q,\mathcal{P},\kappa}| > \min\{|\Omega^1_{q,\mathcal{P},\kappa}|, |\Omega^0_{q,\mathcal{P},\kappa}|\} \text{ and } |\Omega^1_{q,\mathcal{P},\kappa}| > |\Omega^0_{q,\mathcal{P},\kappa}|, \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

Note that same values may be assigned to more than one index κ. In DC, we determine the label ℓ by (4.3) for given q and $\mathcal{P}$.

How to select a partition for each attribute.

For each attribute $q \in \{1, 2, \dots, N\}$, we select a partition $\mathcal{P}$ such that $\chi_q = (q, \mathcal{P}, \ell)$ attains the smallest misclassification cost among all candidates, by which we obtain the set $D = \{\chi_1, \chi_2, \dots, \chi_N\}$ of N discretizers. Note that the search of a partition is independent between attributes because it is based on the domain of one attribute.

For a numerical attribute q, we examine such partitions whose cardinality is not larger than the parameter K (i.e., $|\mathcal{P}| \le K$) in order to save computation time. Then we obtain an optimum partition $\mathcal{P}$ by solving the corresponding dynamic programming [31, 37]: Assuming that there are m distinct values of attribute q in Ω (and thus $m \le |\Omega|$), we write by w_1, w_2, $\dots$, w_m these m distinct values, where $w_1 < w_2 < \cdots < w_m$ holds. For two indices $i, i' \in \{1, 2, \dots, m\}$ $(i \le i')$, let us denote:

$$\Omega(i, i') = \{\omega \in \Omega \mid w_i \le \omega_q \le w_{i'}\},$$

$\Omega^1(i, i') = \Omega^1 \cap \Omega(i, i')$ and $\Omega^0(i, i') = \Omega^0 \cap \Omega(i, i')$. Let us denote by $\mathcal{I}_d(i, i')$ the family of sets of indices such that each member $I = \{i_1, i_2, \dots, i_d\} \in \mathcal{I}_d$ satisfies $i_1 \le i_2 \le \cdots \le i_d$,

$i_1 = i$ and $i_d = i'$. Let us define $\rho_k(\Omega(i,i'))$ as follows:

$$\rho_k(\Omega(i,i')) = \min_{I \in \mathcal{I}_d : 2 \le d \le k+1} \sum_{\kappa=1,2,\ldots,|I|-1} \min\{|\Omega^1(i_\kappa, i_{\kappa+1})|, |\Omega^0(i_\kappa, i_{\kappa+1})|, \mu'|\Omega(i_\kappa, i_{\kappa+1})|\}, \tag{4.4}$$

i.e., $\rho_k(\Omega(i,i'))$ denotes the minimum classification cost over the subset $\Omega(i,i')$ among the partitions of cardinality of at most k. The equation (4.4) is solved by the following dynamic programming:

$$\rho_k(\Omega(i,i')) = \begin{cases} \min\{|\Omega^1(i,i')|, |\Omega^0(i,i')|, \mu'|\Omega(i,i')|\} & \text{if } k = 1, \\ \min\{\rho_{k-1}(\Omega(i,i')), \\ \quad \min_{\iota=i,i+1,\ldots,i'}\{\rho_1(\Omega(i,\iota)) + \rho_{k-1}(\Omega(\iota+1,i'))\}\} & \text{otherwise.} \end{cases}$$

Then the minimum classification cost over the entire data set Ω among the partitions of cardinality of at most K is obtained by solving $\rho_K(1,m)$.

Assume that the minimum cost for $\rho_K(1,m)$ is attained by the index set $I = \{i_1, i_2, \ldots, i_d\}$, where $i_1 = 1$ and $i_d = m$. Then, we adopt the following partition $\mathcal{P}$.

$$\mathcal{P} = \left\{[\min \mathbb{D}_q, \frac{w_{i_2-1} + w_{i_2}}{2}), [\frac{w_{i_2-1} + w_{i_2}}{2}, \frac{w_{i_3-1} + w_{i_3}}{2}), \ldots, [\frac{w_{i_{d-1}-1} + w_{i_{d-1}}}{2}, \max \mathbb{D}_q]\right\}.$$

Note that $|\mathcal{P}| = d - 1 \in [1, K]$ holds, as is observed from (4.4).

For other types of cost functions on numerical attributes, one finds intensive studies on minimization of Daróczy's generalized entropy [23] and Shannon's entropy [32].

For a categorical attribute, we obtain an optimum partition as the family where each element is a singleton of a categorical value. One can easily verify that this is optimum analogously with the correctness of the Naïve-Bayesian approach [28].

4.2.3 Discretization scheme SC

This subsection shows the discretization scheme SC. It constructs a set $D = \{\chi_1, \chi_2, \ldots, \chi_n\}$ of discretizers, where the dimensionality n is determined by our greedy algorithm. SC has two parameters, Γ_{SC} and $\mathbb{V}$. The parameter Γ_{SC} specifies the cost function to evaluate a discretizer set in the greedy algorithm, which is either *data space error* $\Gamma_{\mathrm{SC,ERR}}$ or *unseparated pairs* $\Gamma_{\mathrm{SC,PAIR}}$. The algorithm selects discretizers based on the specified cost function, where the greedy algorithm for $\Gamma_{\mathrm{SC,PAIR}}$ was first proposed by Mii [55]. The other parameter $\mathbb{V} \in \{\mathbb{B}, \mathbb{M}\}$ restricts the cardinality of a partition by $|\mathbb{V}|$ (i.e., 2 or 3) and the range of a label by $\mathbb{V}$, i.e., we consider only such a discretizer $\chi = (q, \mathcal{P}, \ell)$ that satisfies $|\mathcal{P}| = |\mathbb{V}|$ and $\ell : \{1, \ldots, |\mathbb{V}|\} \to \mathbb{V}$.

How to determine a label.

Let us take a partition $\mathcal{P} = \{P_1, \dots, P_{|\mathbb{V}|}\}$ on attribute q. Different from DC, we assign any value of $\mathbb{V}$ to some output value $\ell(\kappa)$ ($\kappa = 1, \dots, |\mathbb{V}|$) in order to distinguish partitioned subsets $\mathbb{D}_q = P_1 \cup \cdots \cup P_{|\mathbb{V}|}$ by ℓ, by which we separate the data space $\mathbb{S} = \mathbb{D}_1 \times \mathbb{D}_2 \times \cdots \times \mathbb{D}_N$ into some subspaces. Under this concept, we determine the label $\ell : \{1, \dots, |\mathbb{V}|\} \to \mathbb{V}$ so that the number of examples which are classified correctly by ℓ (as a classifier with reject option) is maximized. If $\mathbb{V} = \mathbb{B}$, then such ℓ is determined as follows;

$$(\ell(1), \ell(2)) = \begin{cases} (1,0) & \text{if } |\Omega^1_{q,\mathcal{P},1}| + |\Omega^0_{q,\mathcal{P},2}| \geq |\Omega^0_{q,\mathcal{P},1}| + |\Omega^1_{q,\mathcal{P},2}|, \\ (0,1) & \text{otherwise.} \end{cases}$$

Note that $*$ is not used for an output. If $\mathbb{V} = \mathbb{M}$, then we determine ℓ as follows: We set $(\ell(\kappa), \ell(\kappa')) = (1,0)$ for such a pair (κ, κ') that maximizes the sum $|\Omega^1_{q,\mathcal{P},\kappa}| + |\Omega^0_{q,\mathcal{P},\kappa'}|$ among all $\kappa, \kappa' \in \{1,2,3\}, \kappa \neq \kappa'$. We then set $\ell(\kappa'') = *$ to the remaining $\kappa'' = \{1,2,3\} \setminus \{\kappa, \kappa'\}$.

How to select an attribute and a partition.

Let us denote by $D = \{\chi_1, \chi_2, \dots, \chi_n\}$ a set of n discretizers ($n \geq 1$). For a vector $s \in \mathbb{V}^n$, we define a subset $\Omega_{D,s} \subseteq \Omega$ to be $\Omega_{D,s} = \{\omega \in \Omega \mid D(\omega) = s\}$. We write $\Omega^1_{D,s} = \Omega^1 \cap \Omega_{D,s}$ and $\Omega^0_{D,s} = \Omega^0 \cap \Omega_{D,s}$. (Then the data set Ω is partitioned by D as $\Omega = \bigcup_{s \in \mathbb{V}^n} \Omega_{D,s}$.) We then define data space error $\Gamma_{\text{SC,ERR}}(D)$ and unseparated pairs $\Gamma_{\text{SC,PAIR}}(D)$ to be;

$$\Gamma_{\text{SC,ERR}}(D) = \sum_{s \in \mathbb{V}^n} \min\{|\Omega^1_{D,s}|, |\Omega^0_{D,s}|\}, \qquad \Gamma_{\text{SC,PAIR}}(D) = \sum_{s \in \mathbb{V}^n} |\Omega^1_{D,s}| \cdot |\Omega^0_{D,s}|.$$

We define $\Gamma_{\text{SC,ERR}}(\emptyset) = \min\{|\Omega^1|, |\Omega^0|\}$ and $\Gamma_{\text{SC,PAIR}}(\emptyset) = |\Omega^1| \cdot |\Omega^0|$ for convenience. With functions $\Gamma_{\text{SC,ERR}}$ and $\Gamma_{\text{SC,PAIR}}$, we evaluate how D partitions the data space $\mathbb{S}$ into "well-separated" subspaces.

Let Γ_{SC} represent the cost function of either $\Gamma_{\text{SC,ERR}}$ or $\Gamma_{\text{SC,PAIR}}$. Now we describe the greedy algorithm for SC as follows.

Algorithm Γ_{SC}-GREEDY

Input: A data set Ω with N attributes.

Output: A set D of discretizers.

Step 1: Let $D := \emptyset$.

Step 2: Select such χ that minimizes $\Gamma_{\text{SC}}(D \cup \{\chi\})$ among N candidates, each of which is chosen from one attribute.

Step 3: If $\Gamma_{\mathrm{SC}}(D \cup \{\chi\}) < \Gamma_{\mathrm{SC}}(D)$ holds, then let $D := D \cup \{\chi\}$ and return to Step 2. Otherwise, output D and halt.

One can see that $\Gamma_{\mathrm{SC,ERR}}(D')$ is monotone non-increasing, while $\Gamma_{\mathrm{SC,PAIR}}(D')$ is monotone decreasing with respect to the set inclusion over all subsets $D' \subseteq D$. Hence, with $\Gamma_{\mathrm{SC}} = \Gamma_{\mathrm{SC,ERR}}$, the greedy algorithm may halt even if $\Gamma_{\mathrm{SC,ERR}}(D) = 0$ is not attained. On the other hand, with $\Gamma_{\mathrm{SC}} = \Gamma_{\mathrm{SC,PAIR}}$, the algorithm always attains $\Gamma_{\mathrm{SC,PAIR}}(D) = 0$ upon its completion. For a discretizer set D, we note that $\Gamma_{\mathrm{SC,PAIR}}(D) = 0$ holds if and only if $\Gamma_{\mathrm{SC,ERR}}(D) = 0$ holds. Thus use of $\Gamma_{\mathrm{SC}} = \Gamma_{\mathrm{SC,PAIR}}$ may construct a discretizer set including more detailed information on Ω since $\Gamma_{\mathrm{SC}} = \Gamma_{\mathrm{SC,ERR}}$ may output such a discretizer set D'' with $\Gamma_{\mathrm{SC,ERR}}(D'') > 0$ (and thus $\Gamma_{\mathrm{SC,PAIR}}(D'') > 0$). Note that finding a minimum sized D attaining $\Gamma_{\mathrm{SC,PAIR}}(D) = 0$ (and thus $\Gamma_{\mathrm{SC,ERR}}(D) = 0$) is an NP-hard problem [27].

From a numerical attribute, we investigate all possible partitions of cardinality of most $|\mathbb{V}|$, and select the best one as the candidate (note that there are $O(|\Omega|^{|\mathbb{V}|})$ distinct partitions where $|\mathbb{V}| = 2$ or 3). On the other hand, there are $|\mathbb{V}|^m$ possible partitions for a categorical attribute q, where m denotes the number of categories for attribute q and $m = O(|\Omega|)$. Since the size of $|\mathbb{V}|^m$ can be extremely large, we search the partition by a heuristic method based on local search (which is almost equivalent to the discretization scheme by Mii [55]), and use it as the candidate from attribute q.

4.3 Computational experiments

Let us denote by ALG-ICF$^*_{\mathrm{DC}}$ (resp., ALG-ICF$^*_{\mathrm{SC}}$ and ALG-ICF$^*_{\mathrm{IC}}$) the algorithm ALG-ICF* equipped with discretization schemes DC (resp., SC and IC). In this section, we first examine the advantages and defects of algorithms ALG-ICF$^*_{\mathrm{DC}}$ and ALG-ICF$^*_{\mathrm{SC}}$ through computational experiments. Based on this observation, we propose another construction algorithm ALG-ICF$^*_{\mathrm{IC}}$ with a new discretization scheme IC.

Experimental setting.

For the experiments, we use data sets from UCI Repository of Machine Learning [42] as real world data sets. The summary is shown in Table 2.1, where N_{num} (resp., N_{cat}) denotes the number of numerical (resp., categorical) attributes.

Let $\mathcal{C}$ represent a discretization scheme among DC, SC and IC. For a real world data set Ω, we evaluate the performance of the algorithm ALG-ICF$^*_{\mathcal{C}}$ as follows:

(1) We divide Ω into two halves at random, one for the training set Ω_{train} and the other for the test set Ω_{test}.

(2) We construct a discretizer set $D_{\mathcal{C}}$ by applying $\mathcal{C}$ to the training set Ω_{train}, from which we obtain an $\mathbb{M}$-valued training set $X_{\text{train}} = D_{\mathcal{C}}(\Omega_{\text{train}})$.

(3) We construct an ICF classifier f by applying the original ALG-ICF* to X_{train}, and measure its error rate $e(f, X_{\text{test}})$ on an $\mathbb{M}$-valued test set $X_{\text{test}} = D_{\mathcal{C}}(\Omega_{\text{test}})$.

We repeat the process of (1) to (3) 10 times for a set of given parameter values (i.e., α, β, μ, π for ALG-ICF* and ones for discretization scheme $\mathcal{C}$), and we use the average of error rates on test sets as the performance evaluator.

Note that the above experimental setting is different from that of Chapter 3 in the order of division of Ω and discretization. Since we now consider constructing a classifier from real world data sets, $D_{\mathcal{C}}$ should be constructed only from the information of a training set. A test set, an approximation set of future data elements, should not be used in construction of $D_{\mathcal{C}}$. Thus division of the data set is prior to discretization here.

Results on ALG-ICF$^*_{\text{DC}}$ and ALG-ICF$^*_{\text{SC}}$.

Now we show the experimental results in Tables 4.2 and 4.3, where each row and column corresponds to a data set and a construction algorithm, respectively. The parameters used for discretization schemes are written at the top of the table; e.g., as to ALG-ICF$^*_{\text{DC}}$, we show only the result of $K = 3$ and $\mu' = 0.3$ in the table, which was fairly better than all other tested values of $K \in \{2, 3, \dots, 6\}$ and $\mu' \in \{0.1, 0.2, \dots, 0.5\}$.

For each data set, an indicated value in Table 4.2 (resp., Table 4.3) denotes the best (resp., average) true error rate in all combinations of parameters: For ALG-ICF$^*_{\text{DC}}$ and ALG-ICF$^*_{\text{SC}}$, we take $\alpha \in \{0.01, 0.05, 0.1, 0.25, 0.5\}$, $\beta = 0.3$, $\mu \in \{0.1, \dots, 0.5\}$, and $\pi = 0.3$. For C4.5 [65], we use 1%, 5%, 10%, 25%, 50%, 75%, 100% as its confidence level (which was described in Section 2.3.2). Note that we exploit an algorithm that constructs C4.5 decision trees directly from Ω_{train} (not from X_{train}) since this chapter discusses construction algorithms on real world data sets. For ICF algorithms, an error rate smaller than C4.5 is indicated by boldface. For each data set, a sign $\star$ shows the best error rate among all construction algorithms.

The bottom of the tables shows the average of presented error rates for all data sets. The row Best in Table 4.2 represents the average of the best error rate for each data set, among those realized by adjusting parameters as above. On the other hand, the row Avg in Table 4.3 represents the average of error rates observed in all data sets and in all tested parameter

Table 4.2: Best true error rates ($\times 10^2$)

Data	ALG-ICF$^*_{\mathrm{DC}}$	ALG-ICF$^*_{\mathrm{SC}}$				C4.5
	$K = 3$	$\Gamma_{\mathrm{SC}} = \Gamma_{\mathrm{SC,ERR}}$		$\Gamma_{\mathrm{SC}} = \Gamma_{\mathrm{SC,PAIR}}$		
	$\mu' = 0.3$	$\mathbb{V} = \mathbb{B}$	$\mathbb{V} = \mathbb{M}$	$\mathbb{V} = \mathbb{B}$	$\mathbb{V} = \mathbb{M}$	
AUS	**15.42**	**15.21**	**15.68**	**15.42**	$\star$**15.13**	15.95
BCW	$\star$**3.71**	**4.18**	**4.26**	**4.73**	**4.18**	5.00
BUPA	**36.82**	$\star$**34.10**	**36.35**	**36.12**	**35.66**	37.12
CAR	7.02	6.94	6.55	$\star$**1.12**	**1.38**	2.36
CRX	**13.45**	$\star$**13.11**	**13.79**	**13.45**	**13.66**	14.33
FLAG	10.92	$\star$**9.69**	12.47	**10.51**	12.88	10.51
HABER	27.21	27.14	27.61	$\star$**25.91**	26.66	26.27
HEART	$\star$**18.88**	**22.29**	25.77	**19.99**	25.55	24.45
IONO	**11.59**	$\star$**11.42**	12.44	13.86	13.63	12.32
MUSH	$\star$**0.00**	0.23	0.23	**0.01**	$\star$**0.00**	0.01
PIMA	$\star$**24.60**	26.17	27.42	26.64	26.45	25.52
TTT	24.80	8.18	13.84	$\star$**5.26**	13.96	8.08
VOTES	4.35	4.49	4.49	4.77	4.26	$\star$3.98
BEST	15.29	**14.08**	15.45	$\star$**13.67**	14.87	14.30

values. If a construction algorithm Λ outperforms other Λ' in BEST (i.e., Λ achieves a smaller BEST value than Λ'), it means that Λ *can* construct a better classifier than Λ' by tuning up the parameter values. (Note that, however, determining appropriate parameter values is usually difficult.) On the other hand, if Λ outperforms Λ' in AVG, Λ should construct a better classifier by *arbitrary* parameter values. We observe that ALG-ICF$^*_{\mathrm{DC}}$ and ALG-ICF$^*_{\mathrm{SC}}$ outperforms C4.5 in BEST but does not in AVG. In Chapter 3, we showed that ALG-ICF* outperforms C4.5 in BEST on $\mathbb{B}^n$-valued data sets. Thus we see that, with the discretization schemes DC or SC, ALG-ICF* retains its high performance even on real world data sets. In the following, we consider how to improve the discretization schemes so that ICF algorithms has a better performance in AVG.

Recall that BCW, BUPA, HABER, IONO and PIMA consist only of numerical attributes, CAR, MUSH, TTT and VOTES do of categorical attributes, and the rest data sets do of both types of attributes (see Table 2.1). We consider that ALG-ICF$^*_{\mathrm{DC}}$ has a good performance in data sets with numerical attributes partly because an effective discretizer for a numerical attribute can be constructed by dynamic programming.

Table 4.3: Average true error rates ($\times 10^2$)

Data	ALG-ICF$^*_{\text{DC}}$	ALG-ICF$^*_{\text{SC}}$				C4.5
	$K = 3$	$\Gamma_{\text{SC}} = \Gamma_{\text{SC,ERR}}$		$\Gamma_{\text{SC}} = \Gamma_{\text{SC,PAIR}}$		
	$\mu' = 0.3$	$\mathbb{V} = \mathbb{B}$	$\mathbb{V} = \mathbb{M}$	$\mathbb{V} = \mathbb{B}$	$\mathbb{V} = \mathbb{M}$	
AUS	**⋆16.61**	17.7	18.72	18.45	18.15	17.48
BCW	**4.78**	**4.85**	**4.98**	**5.20**	**⋆4.62**	5.21
BUPA	39.90	**⋆36.42**	**37.48**	**37.81**	38.24	38.18
CAR	7.28	8.17	7.76	2.87	**⋆2.68**	2.73
CRX	**⋆14.76**	**16.65**	17.32	17.15	17.13	16.74
FLAG	**12.61**	**⋆11.01**	14.93	14.75	15.10	12.77
HABER	**⋆27.64**	**28.26**	29.31	29.10	29.27	28.40
HEART	**⋆20.42**	**25.27**	28.37	**23.06**	29.44	25.50
IONO	**12.50**	**⋆12.32**	13.53	15.89	16.13	12.64
MUSH	0.02	0.23	0.23	⋆0.01	⋆0.01	⋆0.01
PIMA	**⋆25.75**	28.41	29.43	29.22	29.81	27.90
TTT	26.58	12.11	17.51	9.01	17.34	⋆8.54
VOTES	5.04	5.42	5.42	5.73	5.72	⋆4.66
AVG	16.45	15.90	17.30	16.01	17.20	⋆15.44

For ALG-ICF$^*_{\text{SC}}$, use of $\mathbb{V} = \mathbb{B}$ outperforms $\mathbb{V} = \mathbb{M}$ regardless of Γ_{SC}, as is observed from almost all data sets in both Tables 4.2 and 4.3. We consider that this is because the discretization process with $\mathbb{V} = \mathbb{M}$ partitions the data space into rather too small subspaces and the resulting data set X may include misleading information for classifier construction. Then a classifier constructed from such X rather easily overfit to Ω.

As to evaluation function, use of $\Gamma_{\text{SC}} = \Gamma_{\text{SC,PAIR}}$ is effective particularly for the data sets consisting only of categorical attributes (i.e., CAR, MUSH, TTT and VOTES). It is empirically known that these data sets contain enough information to produce good classifiers. As mentioned in Section 4.2.3, use of $\Gamma_{\text{SC}} = \Gamma_{\text{SC,PAIR}}$ may construct a discretizer set containing more detailed information on Ω than $\Gamma_{\text{SC,ERR}}$, which may explain the above phenomena.

Algorithm ALG-ICF$^*_{\text{IC}}$.

We consider combining plural discretizer sets constructed by different discretization schemes. From the above observation, we introduce a new discretization scheme, integrated construction (IC). Let us denote by $D_{\mathcal{C}}$ a discretizer set constructed by discretization scheme $\mathcal{C}$. Then

Table 4.4: Best true error rates ($\times 10^2$) for various discretizer sets

Data	$D_{\mathrm{DC}(3,0.3)} \cup D_{\mathrm{SC}(\Gamma_{\mathrm{SC}},\mathbb{V})}$				D_{IC}	$D_{\mathrm{IC'}}$	C4.5
	$\Gamma_{\mathrm{SC}} = \Gamma_{\mathrm{SC,ERR}}$		$\Gamma_{\mathrm{SC}} = \Gamma_{\mathrm{SC,PAIR}}$				
	$\mathbb{V} = \mathbb{B}$	$\mathbb{V} = \mathbb{M}$	$\mathbb{V} = \mathbb{B}$	$\mathbb{V} = \mathbb{M}$			
AUS	**15.13**	**15.91**	⋆**15.04**	**15.13**	**15.13**	**15.15**	15.95
BCW	**4.12**	**4.06**	⋆**3.85**	**4.06**	**4.12**	**3.94**	5.00
BUPA	⋆**33.81**	**35.26**	**34.97**	**35.78**	⋆**33.81**	**34.27**	37.12
CAR	2.48	**1.80**	**1.12**	**1.03**	**1.12**	⋆**0.99**	2.36
CRX	⋆**12.87**	**13.82**	⋆**12.87**	**13.85**	⋆**12.87**	**13.33**	14.33
FLAG	⋆**10.00**	11.95	10.61	11.23	⋆**10.00**	10.61	10.51
HABER	27.34	27.21	27.48	26.46	27.34	27.00	⋆26.27
HEART	**20.66**	**20.59**	**19.25**	⋆**18.07**	**20.66**	**20.81**	24.45
IONO	⋆**10.85**	**12.15**	**11.47**	**11.36**	⋆**10.85**	**11.19**	12.32
MUSH	⋆**0.00**	⋆**0.00**	⋆**0.00**	⋆**0.00**	⋆**0.00**	⋆**0.00**	0.01
PIMA	26.04	27.08	**25.46**	25.93	26.04	⋆**25.39**	25.52
TTT	**6.38**	11.33	**5.11**	10.75	**5.11**	⋆**4.44**	8.08
VOTES	4.44	4.44	4.40	4.44	4.40	4.40	⋆3.98
Best	**13.39**	**14.27**	**13.20**	**13.69**	⋆**13.18**	**13.19**	14.30

we integrate DC and SC as follows:

- If the data set consists only of categorical attributes (i.e., $N_{\mathrm{num}} = 0$), then we use $D = D_{\mathrm{DC}(3,0.3)} \cup D_{\mathrm{SC}(\Gamma_{\mathrm{SC,PAIR}},\mathbb{B})}$ as the discretizer set.
- Otherwise, we use $D = D_{\mathrm{DC}(3,0.3)} \cup D_{\mathrm{SC}(\Gamma_{\mathrm{SC,ERR}},\mathbb{B})}$.

Table 4.4 (resp., Table 4.5) shows the best (resp., average) true error rates of classifiers (where we use the same parameter values as Tables 4.2 and 4.3). The tables also show the error rates of other combinations of discretizer sets; how they are combined is shown at the top of tables. ($D_{\mathrm{IC'}}$ is described below.) Note that the error rates of C4.5 are shown again for clarity and thus the values are the same as Tables 4.2 and 4.3).

As seen from the result, ALG-ICF$^*_{\mathrm{IC}}$ outperforms C4.5 not only in Best but also in Avg. This indicates that ALG-ICF$^*_{\mathrm{IC}}$ is better than C4.5 in a stronger sense than the original ALG-ICF* in Chapter 3 is. It is interesting to see that the performance is enhanced by integrating two discretization schemes of different concepts; in other words, either of DC and SC may not provide enough information with ICF learning by itself.

Table 4.5: Average true error rates ($\times 10^2$) for various discretizer sets

Data	$D_{\mathrm{DC}(3,0.3)} \cup D_{\mathrm{SC}(\Gamma_{\mathrm{SC}},\mathbb{V})}$				D_{IC}	$D_{\mathrm{IC}'}$	C4.5
	$\Gamma_{\mathrm{SC}} = \Gamma_{\mathrm{SC,ERR}}$		$\Gamma_{\mathrm{SC}} = \Gamma_{\mathrm{SC,PAIR}}$				
	$\mathbb{V} = \mathbb{B}$	$\mathbb{V} = \mathbb{M}$	$\mathbb{V} = \mathbb{B}$	$\mathbb{V} = \mathbb{M}$			
AUS	⋆**17.35**	18.34	17.52	17.51	⋆**17.35**	17.68	17.48
BCW	⋆**4.63**	**4.64**	**4.71**	**5.00**	⋆**4.63**	**4.68**	5.21
BUPA	**36.27**	**36.97**	**36.98**	38.19	**36.27**	⋆**36.18**	38.18
CAR	3.60	3.24	2.88	⋆**2.18**	2.88	**2.60**	2.73
CRX	⋆**16.46**	16.93	17.03	**16.67**	⋆**16.46**	16.89	16.74
FLAG	⋆**12.30**	14.92	14.56	14.17	⋆**12.30**	14.10	12.77
HABER	⋆**28.17**	29.93	29.37	29.67	⋆**28.17**	28.93	28.40
HEART	**22.89**	**23.22**	⋆**22.54**	**23.30**	**22.89**	**23.14**	25.50
IONO	⋆**12.27**	13.48	12.72	13.05	⋆**12.27**	**12.42**	12.64
MUSH	0.02	0.02	⋆**0.00**	⋆**0.00**	⋆**0.00**	0.02	0.01
PIMA	28.04	28.92	28.39	29.12	28.04	28.05	⋆27.90
TTT	10.43	15.52	9.17	16.14	9.17	9.04	⋆8.54
VOTES	5.43	5.38	5.86	5.60	5.86	5.91	⋆4.66
AVG	**15.22**	16.27	15.51	16.20	⋆**15.09**	**15.35**	15.44

We do not observe that, however, integration of more discretization schemes always enhances ICF classifiers; As shown, if we construct a discretizer set by the scheme IC$'$ as $D_{\mathrm{IC}'} = D_{\mathrm{DC}(3,0.3)} \cup D_{\mathrm{SC}(\Gamma_{\mathrm{SC,ERR}},\mathbb{B})} \cup D_{\mathrm{SC}(\Gamma_{\mathrm{SC,PAIR}},\mathbb{B})}$, the error rates become slightly worse than ALG-ICF$^*_{\mathrm{IC}}$. Also, an integration method of this type increases the size of discretizer set. It can increase the computation time of classifier construction process, as shown in Table 4.6: In Table 4.6, we show the computation time required to construct a classifier in an upper entry and the number of $\mathbb{M}$-valued attributes in a lower entry. (The latter is present only for ICF algorithms.) Note that they are the averaged values in 10 divisions of Ω into training and test sets. A sign - indicates the time smaller than 0.01 seconds. The computation time for ICF algorithms consists of the time for constructing a discretizer set and that for ALG-ICF*. We observe that the latter requires much more time than the former in almost all cases.

4.4 Concluding remarks

In this chapter, we considered how to extend ICF classifiers, originally proposed on $\mathbb{M}$-valued data sets, so as to handle real world data sets. In order to process such data sets by ICF, we apply a discretization scheme to the given data set, and construct a classifier from the discretized data set. We first introduced two discretization schemes DC and SC, and proposed a new one, IC, based on the experimental results on the formers. We observed that ALG-ICF$^*_{\text{IC}}$ outperforms C4.5 in a stronger sense than Chapter 3.

For future work, an alternative approach to ICF extension is to enable the original ICF (which is defined on $\mathbb{S} = \mathbb{M}^n$) to handle general discrete data sets, i.e., $\mathbb{S} = \mathbb{D}_1 \times \mathbb{D}_2 \times \dots \mathbb{D}_N$, where each $\mathbb{D}_q$ $(q = 1, 2, \dots, N)$ is a set of discrete values rather than 0, 1 and $*$. By this, we may be able to concentrate on numerical attributes in the discretization process.

Table 4.6: Computation time (sec.) for constructing a classifier (upper) and the number of $\mathbb{M}$-valued attributes for ICF algorithms (lower)

Data	ALG-ICF$^*_{\text{DC}}$	ALG-ICF$^*_{\text{SC}}$				ALG-	ALG-	C4.5
	$K = 3$	$\Gamma_{\text{SC}} = \Gamma_{\text{SC,ERR}}$		$\Gamma_{\text{SC}} = \Gamma_{\text{SC,PAIR}}$		ICF$^*_{\text{IC}}$	ICF$^*_{\text{IC}'}$	
	$\mu' = 0.3$	$\mathbb{V} = \mathbb{B}$	$\mathbb{V} = \mathbb{M}$	$\mathbb{V} = \mathbb{B}$	$\mathbb{V} = \mathbb{M}$			
AUS	0.66	8.83	6.91	4.68	5.00	10.50	18.25	0.03
	14.0	18.7	11.4	13.2	8.6	32.7	45.9	
BCW	0.08	0.19	0.10	0.26	0.08	0.19	0.38	-
	9.0	8.3	6.9	9.0	5.8	17.3	26.3	
BUPA	0.06	3.77	1.30	3.83	1.04	5.33	9.66	0.01
	6.0	16.9	9.8	15.1	8.3	22.9	38.0	
CAR	0.20	6.04	1.73	7.04	0.94	12.52	22.83	-
	6.0	9.3	8.7	12.0	7.0	15.3	27.3	
CRX	0.62	5.91	5.28	3.03	3.78	7.53	14.83	0.03
	15.0	16.0	10.4	11.8	8.0	31.0	42.8	
FLAG	0.09	0.05	0.06	0.06	0.06	0.28	0.55	-
	28.0	6.2	6.3	7.3	5.0	34.2	41.5	
HABER	-	1.19	0.45	4.01	0.47	1.59	6.25	-
	3.0	9.3	11.6	20.2	10.5	14.6	25.1	
HEART	0.25	0.55	0.42	0.27	0.33	0.80	1.30	-
	13.0	11.2	8.3	9.1	6.4	24.2	33.3	
IONO	1.10	0.20	4.51	0.28	4.13	1.34	6.00	0.05
	34.0	8.7	6.0	8.6	5.2	42.7	51.3	
MUSH	7.34	0.55	0.55	0.30	0.17	9.18	15.07	0.05
	22.0	4.6	4.6	5.0	4.0	26.6	31.6	
PIMA	1.14	41.47	21.21	32.77	17.30	50.78	78.51	0.03
	8.0	20.4	12.0	16.0	8.9	28.4	44.4	
TTT	2.27	5.62	5.20	4.93	5.00	10.16	17.49	-
	9.0	9.8	8.7	10.1	7.7	18.8	28.9	
VOTES	0.09	0.05	0.05	0.08	0.06	0.20	0.26	-
	16.0	4.7	4.7	7.9	7.2	20.7	28.6	

Chapter 5

A Randomness Based Analysis on the Data Size Needed for Removing Deceptive Rules

5.1 Introduction

Recall that the approach to a general learning problem consists of the four steps described in Section 1.1, where we mentioned the importance of the data size for acquiring meaningful knowledge. A large data set should provide us with enough information to acquire knowledge that describes the hidden structure of data successfully. On the other hand, we expect that structure acquired from a small data set may be deceptive or useless in many cases.

In classification problem, it is known that the VC dimension of a representation model gives a necessary or sufficient data size such that the empirical error rate and true error rate become close to each other with a high probability [14, 30, 78]. Unfortunately, we conjecture that it is difficult to compute the VC dimension of an ICF classifier with a fixed DAG exactly since it is shown that the same problem on NN, where a classifier is also formulated by DAG, is already difficult [6, 11, 12, 53].

In this chapter, we concentrate on more tractable structure of knowledge, called a pattern, and discuss the data size necessary and sufficient for learning patterns successfully. We assume the data space to be $\mathbb{S} = \mathbb{B}^n$ again throughout this chapter. A pattern $z = (J, b)$ is defined by a set $J \subseteq \{1, 2, \ldots, n\}$ of n indices and a $|J|$-dimensional binary vector $b \in \mathbb{B}^J$. For a data element $x \in \mathbb{B}^n$, we say that z *covers* x if $x|_J = b$ holds. We denote by $\mathbb{B}^n(z)$ the set of all data elements covered by z, i.e., $\mathbb{B}^n(z) = \{x \in \mathbb{B}^n \mid x|_J = b\}$. For a data set $X = X^1 \cup X^0$, let us denote by $X(z)$ the set of examples in X covered by z, i.e., $X(z) = X \cap \mathbb{B}^n(z)$. We define the *frequency* of z in X by:

$$\lambda(z, X) = \frac{|X(z)|}{|X|}.$$

For a given constant $\theta \in [0, 1]$, we say that z is *θ-frequent* (resp., *θ-infrequent*) in X, if $\lambda(z, X) \geq \theta$ (resp., $\lambda(z, X) \leq \theta$) holds. For given constants $\theta_1, \theta_0 \in [0, 1]$, we say that z is a (θ_1, θ_0)-pattern in X, if $\lambda(z, X^1) \geq \theta_1$ and $\lambda(z, X^0) \leq \theta_0$. If θ_1 is "large enough" and θ_0 is "small enough," a (θ_1, θ_0)-pattern represents a trend of attribute values toward X^1 rather than X^0, and thus might be utilized as useful information of data. However, if the size of X is small, a (θ_1, θ_0)-pattern z may be *deceptive*: z may not capture the real trend of data and may not serve as good knowledge.

In this chapter, we consider necessary and sufficient sizes of a data set with which it contains such deceptive patterns with a low probability. In Section 5.2, we define a deceptive (θ_1, θ_0)-pattern, and make an assumption on the probability distribution of data elements. Any data set in this chapter is distributed based on the assumption, which is summarized as Assumption 5.1. This assumption is not a particular one, and is often used in the literature [67].

In Section 5.3, we derive a sufficient data size with which a data set contains deceptive patterns with a low probability. The derivation of the sufficient data size is based on Hoeffding's inequalities [43]. In Section 5.4, we propose a necessary data size by utilizing an observation arising from randomness. Let us call a data set generated at random a *random data set*. If the size is sufficiently large, a random data set does not contain (θ_1, θ_0)-patterns (for a "large" θ_1 and a "small" θ_0). However, if the size is sufficiently small, even a random data set contain (θ_1, θ_0)-patterns, which should be deceptive. Then we claim that any data set should have a certain number of examples such that a random data set with the same number contains (θ_1, θ_0)-patterns with a low probability so as to remove such deceptive (θ_1, θ_0)-patterns. We justify our claim by computational experiments on UCI data sets. We also derive an upper bound on the proposed necessary data size as its estimate.

Notes on patterns.

One may be interested in a trend of attribute values towards X^0; i.e., a pattern z satisfying $\lambda(z, X^1) \leq \theta_1$ and $\lambda(z, X^0) \geq \theta_0$ for "small" θ_1 and "large" θ_0. Since the results in this chapter can be obtained by interchanging the roles of true and false examples, we focus on good patterns for true examples.

Enumerating frequent/infrequent patterns in the data set is an important issue in data mining and bio-informatics (e.g., knowledge discovery from genome databases) [1, 33, 83]. The term "frequent/infrequent set" is widely used in the literature to denote a frequent/infrequent pattern, but in order to distinguish this from a simple set of elements, we use the term "pattern" to denote $z = (J, b)$ in this chapter. It is well-known that one can find frequent/ infrequent patterns in incrementally polynomial time [1], and many fast algorithms for this task have been proposed so far (e.g., [76]). By applying these algorithms, we can enumerate all (θ_1, θ_0)-patterns in the data set X in incrementally polynomial time; e.g., by taking the intersection of the set of θ_1-frequent patterns in X^1 and that of θ_0-infrequent patterns in X^0, both of which can be enumerated in incrementally polynomial time.

A pattern $z = (J, b)$ is called a *maximal frequent pattern* if z is frequent in X and no pattern $z' = (J', b')$ with $J' \supset J$ and $b'|_J = b$ is frequent in X. Also, z is called a *minimal infrequent pattern* if z is infrequent in X and no pattern $z' = (J', b')$ with $J' \subset J$ and $b|_{J'} = b'$ is infrequent in X. Boros et al. [18] showed that, given a family of $O(n^{\varepsilon})$ maximal frequent patterns, it is NP-complete to decide whether X has any other maximal frequent patterns (for arbitrarily small fixed $\varepsilon > 0$), and that all minimal infrequent patterns can be enumerated in incremental quasi-polynomial time.

Related works.

We here describe related works and the main difference between our approach and those existing ones. The problem of enumerating frequent patterns is closely related to that of *association rules*. An association rule is generally defined by a pair of patterns $(z, z') = ((J, b), (J', b'))$ with $J \cap J' = \emptyset$; it represents that an example x with $x|_J = b$ is likely to attain $x|_{J'} = b'$. Patterns in this chapter may be regarded as special cases of association rules such that the classes of examples are attached to the original data set as the $(n + 1)$ st Boolean variable and z' is restricted to $z' = (\{n + 1\}, (1))$.

An association rule (z, z') is usually evaluated by its *support* $\lambda_{\text{sup}}(z, z')$ and *confidence* $\lambda_{\text{conf}}(z, z')$ defined as follows:

$$\lambda_{\text{sup}}(z, z') = \lambda((J \cup J', (b, b')), X) = \frac{|\{x \in X \mid x|_{J \cup J'} = (b, b')\}|}{|X|},$$
$$\lambda_{\text{conf}}(z, z') = \lambda(z', X(r)) = \frac{|\{x \in X(z) \mid x|_{J'} = b'\}|}{|X(z)|},$$

while we evaluate a pattern z by its frequency in X^1 and infrequency in X^0. Thus enumeration of frequent patterns is a basic operation in finding association rules.

As the task of enumerating association rules from a huge data set is very time-consuming, Li and Gopalan [51] and Toivonen [75] discussed the proper size of a randomly drawn subset X' of the original data set X such that $\lambda(z, X')$ is close enough to $\lambda(z, X)$ with a high probability for all patterns z. While they consider random sampling of a manageable size from the given huge data set, we consider the situation in which the size of the given data set is small, and discuss whether the data set contains deceptive patterns or not.

5.2 Preliminaries

Let us define a *data source* by $\mathcal{D} = (\zeta, P^1, P^0)$, where $\zeta \in [0,1]$ denotes a probability, and $P^1, P^0 : \mathbb{B}^n \to [0,1]$ denote probability distributions. Since P^1 and P^0 are probability distributions, it holds that

$$\sum_{x \in \mathbb{B}^n} P^1(x) = \sum_{x \in \mathbb{B}^n} P^0(x) = 1. \tag{5.1}$$

Then we make an assumption on generation of examples as follows.

Assumption 5.1 *For a given data source $\mathcal{D} = (\zeta, P^1, P^0)$, an example x with class $y(x) = C$ is generated by the following steps independently:*

Step 1: *The class C is set to 1 with probability ζ, and to 0 otherwise (i.e., with probability $1 - \zeta$).*

Step 2: *A binary vector x with class $y(x) = C$ is drawn according to the distribution P_C.*

Let us take a data set $X = X^1 \cup X^0$, which is a set of examples generated as above. Since the oracle y in this case is probabilistic, it may occur $X^1 \cap X^0 \neq \emptyset$. If $m_1 = |X^1|$ and $m_0 = |X^0|$, then we call X an (m_1, m_0)-data set.

Let us denote by $P_U : \mathbb{B}^n \to [0,1]$ the uniform distribution on the n-dimensional hyper cube such that $P_U(x) = 1/2^n$ for any $x \in \mathbb{B}^n$. We define a *random data source* by $\mathcal{D}_{\text{rand}} = (\zeta, P_U, P_U)$ for an arbitrary $\zeta \in [0,1]$. We call a data set generated by $\mathcal{D}_{\text{rand}}$ a random data set.

Let us take a data source $\mathcal{D} = (\zeta, P^1, P^0)$ and a pattern $z = (J, b)$. Assumption 5.1 determines the posterior probability $\lambda_1^*(z, \mathcal{D})$ (resp., $\lambda_0^*(z, \mathcal{D})$) with which z covers an example x under the condition that x is a true (resp., false) example as follows:

$$\begin{aligned} \lambda_1^*(z, \mathcal{D}) &= \Pr(x|_J = b \mid y(x) = 1) = \sum_{x \in \mathbb{B}^n(z)} P^1(x), \\ \lambda_0^*(z, \mathcal{D}) &= \Pr(x|_J = b \mid y(x) = 0) = \sum_{x \in \mathbb{B}^n(z)} P^0(x). \end{aligned} \tag{5.2}$$

Note that $\lambda_1^*(z, \mathcal{D})$ (resp., $\lambda_0^*(z, \mathcal{D})$) becomes equivalent with the frequency of z in a large amount of true (resp., false) examples. Let us say that a pattern z is θ_1*-frequent in the true* *class* if $\lambda_1^*(z, \mathcal{D}) \geq \theta_1$ is satisfied. Similarly, we say that z is θ_0-*infrequent in the false class* if $\lambda_0^*(z, \mathcal{D}) \leq \theta_0$ is satisfied. Now we define a (θ_1, θ_0)-deceptive pattern as follows.

Definition 5.1 *Assume that we are given a data source $\mathcal{D} = (\zeta, P^1, P^0)$, a data set X generated by $\mathcal{D}$, constants $\theta_1, \theta_0 \in [0, 1]$. We call a pattern z a deceptive (θ_1, θ_0)-pattern if z is a (θ_1, θ_0)-pattern on X but is not both θ_1-frequent in the true class and θ_0-infrequent in the false class.*

Assume that we are given m_1 true examples generated by $\mathcal{D}$. Then the probability with which a pattern z is θ_1-frequent in these m_1 true examples is determined by:

$$B_+(m_1, \theta_1, \lambda_1^*(z, \mathcal{D})) = \sum_{i=\lceil \theta_1 m_1 \rceil}^{m_1} \binom{m_1}{i} \lambda_1^*(z, \mathcal{D})^i (1 - \lambda_1^*(z, \mathcal{D}))^{m_1 - i}. \tag{5.3}$$

Similarly, if we are given m_0 false examples generated by $\mathcal{D}$, then the probability with which z is θ_0-infrequent in the m_0 false examples is determined by:

$$B_-(m_0, \theta_0, \lambda_0^*(z, \mathcal{D})) = \sum_{i=0}^{\lfloor \theta_0 m_0 \rfloor} \lambda_0^*(z, \mathcal{D})^i (1 - \lambda_0^*(z, \mathcal{D}))^{m_0 - i}. \tag{5.4}$$

For B_+ and B_- in (5.3) and (5.4), the following result was given by Hoeffding [43].

Theorem 5.1 (*Hoeffding* [43]) *Assume that we are given a positive integer m and $\theta \in [0, 1]$. For $p \in [0, \theta]$, B_+ in* (5.3) *satisfies:*

$$B_+(m, \theta, p) \leq \exp(-2m(\theta - p)^2). \tag{5.5}$$

For $p \in [\theta, 1]$, B_- in (5.4) *satisfies:*

$$B_-(m, \theta, p) \leq \exp(-2m(p - \theta)^2). \tag{5.6}$$

5.3 A sufficient data size by Hoeffding's inequalities

By using Hoeffding's inequalities in Theorem 5.1, we derive a sufficient data size with which any data set contains deceptive (θ_1, θ_0)-patterns with a low probability. For this, we introduce two parameters $\varepsilon, \delta \in (0, 1]$, where ε (resp., δ) is a parameter for frequency (resp., probability). Then the following Theorem 5.2 (resp., Theorem 5.3) states that, if a given set X^1 (resp., X^0) of true (resp., false) examples is large, then some of the patterns which are not $(1-\varepsilon)\theta_1$-frequent in the true class (resp., $(1+\varepsilon)\theta_0$-infrequent in the false class) become θ_1-frequent in X^1 (resp., θ_0-infrequent in X^0) with a probability lower than δ.

Theorem 5.2 *Assume that we are given a data source $\mathcal{D}$, a set X^1 of m_1 true examples generated by $\mathcal{D}$, a constant $\theta_1 \in [0, 1]$ and parameters $\varepsilon, \delta \in (0, 1]$. Then some of the patterns which are not $(1-\varepsilon)\theta_1$-frequent in the true class become θ_1-frequent in X^1 with a probability lower than δ if the m_1 satisfies the following:*

$$m_1 > \frac{n \ln 3 - \ln \delta}{2\varepsilon^2\theta_1^2}. \tag{5.7}$$

Proof: Let us assume a pattern z which is not $(1-\varepsilon)\theta_1$-frequent in the true class. Then we have $\lambda_1^*(z, \mathcal{D}) < (1-\varepsilon)\theta_1 < \theta_1$. From (5.5) of Theorem 5.1, the probability with which z is θ_1-frequent in X^1 is bounded as follows:

$$B_+(m_1, \theta_1, \lambda_1^*(z, \mathcal{D})) < B_+(m_1, \theta_1, (1-\varepsilon)\theta_1) < \exp(-2m_1\varepsilon^2\theta_1^2). \tag{5.8}$$

Since there are 3^n patterns in all, some of the patterns which are not $(1-\varepsilon)\theta_1$-frequent in the true class become θ_1-frequent in X^1 with probability at most $3^n \cdot \exp(-2m_1\varepsilon^2\theta_1^2)$. Then the inequality $3^n \cdot \exp(-2m_1\varepsilon^2\theta_1^2) < \delta$ is equivalent with (5.7). $\square$

Theorem 5.3 *Assume that we are given a data source $\mathcal{D}$, a set X^0 of m_0 false examples generated by $\mathcal{D}$, a constant $\theta_0 \in [0, 1]$ and parameters $\varepsilon, \delta \in (0, 1]$. Then some of the patterns which are not $(1+\varepsilon)\theta_0$-infrequent in the false class become θ_0-infrequent in X^0 with a probability lower than δ if the m_0 satisfies the following:*

$$m_0 > \frac{n \ln 3 - \ln \delta}{2\varepsilon^2\theta_0^2}. \tag{5.9}$$

Proof: The proof is similar to Theorem 5.2. Let us assume a pattern z which is not $(1+\varepsilon)\theta_0$-infrequent in the false class. Then from (5.6) of Theorem 5.1,

$$B_-(m_0, \theta_0, \lambda_0^*(z, \mathcal{D})) < B_-(m_0, \theta_0, (1+\varepsilon)\theta_0) < \exp(-2m_0\varepsilon^2\theta_0^2). \tag{5.10}$$

Then the inequality $3^n \cdot \exp(-2m_0\varepsilon^2\theta_0^2) < \delta$ is equivalent with (5.9). $\square$

We finally obtain the following corollary that describes a data size sufficient for removing deceptive (θ_1, θ_0)-patterns of Definition 5.1.

Corollary 5.1 *Assume that we are given an (m_1, m_0)-data set X. If m_1 and m_0 satisfy (5.7) and (5.9) respectively, the probability with which some of the (θ_1, θ_0)-patterns on X become deceptive is lower than 2δ.*

5.4 A necessary data size based on randomness based claim

5.4.1 Randomness based claim

Since a (θ_1, θ_0)-pattern with a "large" θ_1 and a "small" θ_0 should represent a trend of attribute values towards true examples, we assume $\theta_1 > \theta_0$ throughout this section.

A random data set of a sufficiently large size should not contain (θ_1, θ_0)-patterns since any pattern z satisfies $\lambda_1^*(z, \mathcal{D}) = \lambda_0^*(z, \mathcal{D})$ and thus z cannot be θ_1-frequent in the true class and θ_0-infrequent in the false class at the same time. However, if the size is small, even a random data set may contain (θ_1, θ_0)-patterns (all of which are deceptive of Definition 5.1). Based on this observation, we give a claim on deceptive patterns as follows.

Claim 5.1 *Assume that we are given a data source $\mathcal{D}$, an (m_1, m_0)-data set X generated by $\mathcal{D}$ and constants $\theta_1, \theta_0 \in [0, 1]$ $(\theta_1 > \theta_0)$. If a random (m_1, m_0)-data set contains (θ_1, θ_0)-patterns with a high probability, then we claim that X should contain deceptive (θ_1, θ_0)-patterns.*

In other words, in order to remove deceptive (θ_1, θ_0)-patterns from any data set, the data size needs to large enough so that a random data set contains (θ_1, θ_0)-patterns with a low probability.

We conducted the following computational experiments to validate Claim 5.1. Let us denote by $E_{\mathcal{D}}(m_1, m_0; \theta_1, \theta_0)$ the expectation of the number of (θ_1, θ_0)-patterns existing in an (m_1, m_0)-data set generated by data source $\mathcal{D}$, which is determined as follows: For a pattern z, the probability with which z becomes θ_1-frequent in m_1 true examples (resp., θ_0-infrequent in m_0 false examples) is determined by $B_+ = B_+(m_1, \theta_1, \lambda_1^*(z, \mathcal{D}))$ (resp., $B_- = B_-(m_0, \theta_0, \lambda_0^*(z, \mathcal{D}))$), as described in (5.3) and (5.4). Thus z is a (θ_1, θ_0)-pattern in an (m_1, m_0)-data set with probability B_+B_- since examples are assumed to be generated independently. From the linearity of expectation, we have

$$\begin{aligned} E_{\mathcal{D}}(m_1, m_0; \theta_1, \theta_0) &= \sum_{z \in Z} 1 \cdot B_+(m_1, \theta_1, \lambda_1^*(z, \mathcal{D})) \cdot B_-(m_0, \theta_0, \lambda_0^*(z, \mathcal{D})) \\ &\quad + 0 \cdot (1 - B_+(m_1, \theta_1, \lambda_1^*(z, \mathcal{D})) \cdot B_-(m_0, \theta_0, \lambda_0^*(z, \mathcal{D}))) \\ &= \sum_{z \in Z} B_+(m_1, \theta_1, \lambda_1^*(z, \mathcal{D})) \cdot B_-(m_0, \theta_0, \lambda_0^*(z, \mathcal{D})), \end{aligned} \tag{5.11}$$

where Z denotes the set of all patterns (and hence $|Z| = 3^n$ holds).

Let us denote by M_1 and M_0 sufficiently large numbers. Regardless of data source $\mathcal{D}$, $E_{\mathcal{D}}(m_1, m_0; \theta_1, \theta_0)$ should change as follows: If m_1 and m_0 are small, then $E_{\mathcal{D}}(m_1, m_0; \theta_1, \theta_0)$ should be much larger than $E_{\mathcal{D}}(m_1, m_0; \theta_1, \theta_0)$ since every pattern has a large probability

B_+B_- to be a (θ_1, θ_0)-pattern. On the other hand, if m_1 and m_0 are large, it holds that $E_{\mathcal{D}}(m_1, m_0; \theta_1, \theta_0) \simeq E_{\mathcal{D}}(M_1, M_0; \theta_1, \theta_0)$.

We can compute the exact value of the expectation $E_{\mathcal{D}_{\text{rand}}}(m_1, m_0; \theta_1, \theta_0)$ for the random data source $\mathcal{D}_{\text{rand}}$ since $\lambda_1^*(z, \mathcal{D}_{\text{rand}})$ and $\lambda_0^*(z, \mathcal{D}_{\text{rand}})$ are available for any pattern z from (5.2): For a pattern $z = (J, b)$, let us call the cardinality $|J|$ the *level* of z. If the level is k, one can easily see that $\lambda_1^*(z, \mathcal{D}_{\text{rand}}) = \lambda_0^*(z, \mathcal{D}_{\text{rand}}) = 1/2^{n-k}$ holds. Then we decide whether a random (m_1, m_0)-data set contains (θ_1, θ_0)-patterns with a high probability or not based on the size of $E_{\mathcal{D}_{\text{rand}}}(m_1, m_0; \theta_1, \theta_0)$. If the expectation is smaller (resp., larger) than some threshold, we conclude that a random (m_1, m_0)-data set does not contain (resp., contains) (θ_1, θ_0)-patterns with a high probability. In the experiments, we set the threshold to 1.

For an (m_1, m_0)-data set generated by a general data domain $\mathcal{D}$, we decide whether it contains many deceptive (θ_1, θ_0)-patterns or not based on the change of $E_{\mathcal{D}}(m_1, m_0; \theta_1, \theta_0)$; if $E_{\mathcal{D}}(m_1, m_0; \theta_1, \theta_0) \gg E_{\mathcal{D}}(M_1, M_0; \theta_1, \theta_0)$ (resp., $E_{\mathcal{D}}(m_1, m_0; \theta_1, \theta_0) \simeq E_{\mathcal{D}}(M_1, M_0; \theta_1, \theta_0)$), then we consider that an (m_1, m_0)-data set contains many (resp., few) deceptive (θ_1, θ_0)-patterns.

For a real binary data set, however, we cannot compute $E_{\mathcal{D}}(m_1, m_0; \theta_1, \theta_0)$ exactly since the probability distributions P^1, P^0 of $\mathcal{D} = (\zeta, P^1, P^0)$ are unavailable to us. Thus in the experiments, we estimate an expectation by bootstrapping method [29, 81]: Let us denote by $X = X^1 \cup X^0$ the given binary data set. For given numbers m_1 and m_0, we generate an (m_1, m_0)-data set X' by sampling examples from X with replacement and enumerate (θ_1, θ_0)-patterns from X'. We repeat this 100 times and estimate $E_{\mathcal{D}}(m_1, m_0; \theta_1, \theta_0)$ by the average of observed numbers of (θ_1, θ_0)-patterns.

Expectation $E_{\mathcal{D}_{\text{rand}}}$ in random data sets.

We first compute the expectation $E_{\mathcal{D}_{\text{rand}}}(m_1, m_0; \theta_1, \theta_0)$ of the number of (θ_1, θ_0)-patterns in a random (m_1, m_0)-data set by (5.11) with $\mathcal{D} = \mathcal{D}_{\text{rand}}$. In order to compare m_1 and m_0 satisfying $E_{\mathcal{D}_{\text{rand}}}(m_1, m_0; \theta_1, \theta_0) \leq 1$ with the change of $E_{\mathcal{D}}$ for a UCI data set $X = X^1 \cup X^0$ of Table 2.1 later, we adopt the same number of attributes n and the same rate of true and false examples $m_1/m_0 = |X^1|/|X^0|$ as X. For θ_1 and θ_0, we test $\theta_1 \in \{0.10, 0.20\}$ and $\theta_0 \in \{0.00, 0.01, 0.02, 0.05\}$.

Figures 5.1 and 5.2 show the computed expectations with the parameters corresponding to BCW and BUPA, respectively; i.e., $n = 13$ and $m_1 + m_0$ is changed with keeping $m_1/m_0 = 239/444$ for BCW, and $n = 21$ and $m_1/m_0 = 200/145$ for BUPA. Each figure contains two cases corresponding to $\theta_1 = 0.10$ and 0.20, where the horizontal (resp., vertical) axis represents $m_1 + m_0$ (resp., $E_{\mathcal{D}_{\text{rand}}}$) and four curves correspond to different values of θ_0. Note

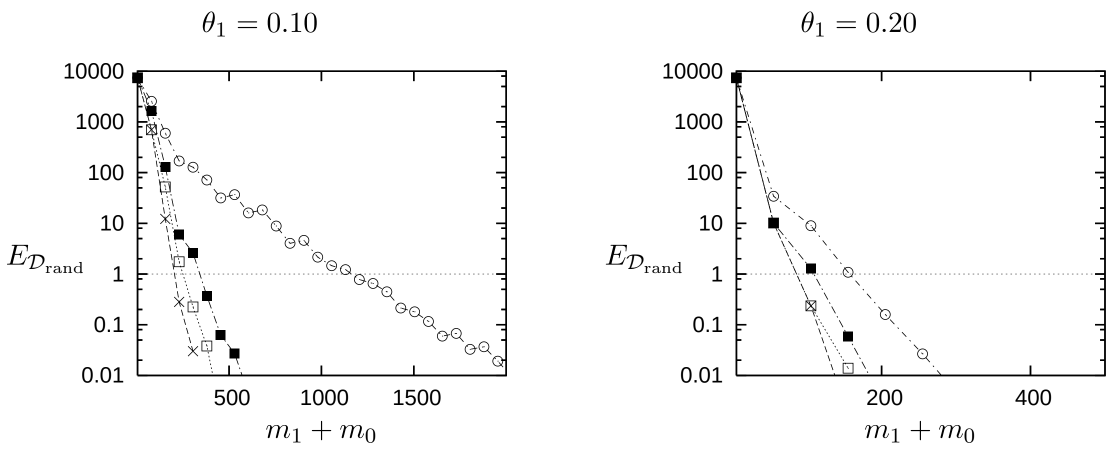


Figure 5.1: Expectation of the number of (θ_1, θ_0)-patterns in random data sets compared to BCW

($n = 12$, $m_1/m_0 = 239/444$. Lines with points $\times, \square, ■, \circ$ represent $\theta_0 = 0.00, 0.01, 0.02, 0.05$, respectively.)

that the vertical axis is in the logarithmic scale. The $E_{\mathcal{D}_{\mathrm{rand}}}$ appears to be monotonically decreasing with m_1+m_0 if we neglect small irregularities, and becomes less than 1 as m_1+m_0 becomes larger than a certain point.

Among the examined values of m_1 (resp., m_0), let us denote by M_1^* (resp., M_0^*) the smallest value that attains $E_{\mathcal{D}_{\mathrm{rand}}}(m_1, m_0; \theta_1, \theta_0) \le 1$. Thus M_1^* (resp., M_0^*) indicates the necessary size of true (resp., false) examples in view of Claim 5.1. Table 5.1 shows the observed $M_1^* + M_0^*$ for UCI data sets.

Expectation $E_{\mathcal{D}}$ in UCI data sets.

We show the (estimated) expectations of the numbers of (θ_1, θ_0)-patterns in UCI data sets in Figures 5.3 and 5.4, where the former is for AUS, BCW, BUPA, CAR, CRX and FLAG, and the latter is for HEART, IONO, MUSH, TTT and VOTES. (Since our experimental scheme utilizing bootstrapping method must require too much computation time for HABER and PIMA, we have not conducted experiments on these data sets.) In these figures, we use $\theta_1 = 0.10$. A broken line parallel to the vertical axis represents $M_1^* + M_0^*$, the proposed necessary data size, which corresponds to the value presented in Table 5.1.

Our claim asserts that m_1+m_0 should be larger than at least $M_1^*+M_0^*$ so that $E_{\mathcal{D}}(m_1, m_0; \theta_1, \theta_0)$ is saturated; indeed, $E_{\mathcal{D}}(M_1^*, M_0^*; \theta_1, \theta_0) \gg E_{\mathcal{D}}(M_1, M_0; \theta_1, \theta_0)$ holds in many cases, which means that an (M_1^*, M_0^*)-data set generated by $\mathcal{D}$ should contain many deceptive

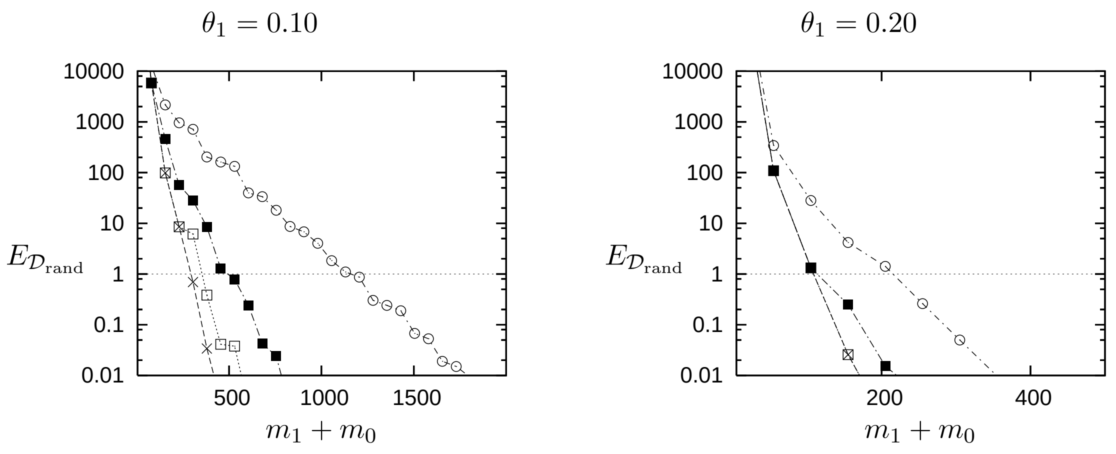


Figure 5.2: Expectation of the number of (θ_1, θ_0)-patterns in random data sets compared to BUPA

($n = 20$, $m_1/m_0 = 200/145$. Lines with points $\times, \square, \blacksquare, \circ$ represent $\theta_0 = 0.00, 0.01, 0.02, 0.05$, respectively.)

(θ_1, θ_0)-patterns, and thus more examples are needed to remove them.

5.4.2 Upper bounds on the necessary data size

We denote by Z_k the set of all possible patterns of level k $(1 \leq k \leq n)$. Note that $|Z_k| = 2^k\binom{n}{k}$ holds and that $|\mathbb{B}^n(z)| = 2^{n-k}$ holds for any $z \in Z_k$. Let and $E_{\mathcal{D},k}(m_1, m_0; \theta_1, \theta_0)$ be the expectation of the number of (θ_1, θ_0)-patterns in an (m_1, m_0)-data set by $\mathcal{D}$ when their levels are restricted to k. Then (5.11) is rewritten as follows:

$$\begin{aligned} E_{\mathcal{D}}(m_1, m_0; \theta_1, \theta_0) &= \sum_{k=1}^{n} E_{\mathcal{D},k}(m_1, m_0; \theta_1, \theta_0) \\ &= \sum_{k=1}^{n} \sum_{z \in Z_k} B_+(m_1, \theta_1, \lambda_1^*(z; \mathcal{D})) B_0(m_0, \theta_0, \lambda_0^*(z; \mathcal{D})). \end{aligned} \tag{5.12}$$

The determination of the proposed necessary data size $M_1^* + M_0^*$ by using (5.11) or (5.12) requires expensive computational time. To alleviate this, we derive upper bounds on M_1^* and M_0^* in this subsection. For the derivation, we assume that any data source $\mathcal{D} = (\zeta, P^1, P^0)$ satisfies the following assumption.

Assumption 5.2 *For any $x \in \mathbb{B}^n$, $P^1(x) \leq p_1$ and $P^0(x) \geq p_0$ hold for some constants p_1 and p_0.*

From (5.1), it is implied that $p_1 \geq 1/2^n$ and $p_0 \leq 1/2^n$. Note that the random data source $\mathcal{D}_{\rm rand}$ is realized by setting $p_1 = p_0 = 1/2^n$.

Table 5.1: The proposed necessary sizes $M_1^* + M_0^*$ for UCI data sets

Data	$\theta_1 = 0.10$				$\theta_1 = 0.20$			
	$\theta_0 = 0.00$	0.01	0.02	0.05	$\theta_0 = 0.00$	0.01	0.02	0.05
AUS	270	315	425	1260	100	100	125	180
BCW	200	255	345	1145	85	85	115	160
BUPA	295	365	465	1120	115	115	115	190
CAR	235	265	400	1265	100	100	115	165
CRX	245	285	395	1125	100	100	110	165
FLAG	195	195	290	740	80	80	80	115
HABER	390	545	785	2430	155	175	195	310
HEART	205	250	340	970	90	90	100	135
IONO	235	235	360	705	95	95	95	140
MUSH	85	85	85	230	40	40	40	65
PIMA	285	400	515	1690	130	130	145	215
TTT	245	245	370	735	100	100	100	145
VOTES	230	230	345	720	90	90	90	140

For a data source $\mathcal{D}$ satisfying Assumption 5.2, we show that an upper bound on $E_{\mathcal{D},k}(m_1, m_0; \theta_1, \theta_0)$ becomes sufficiently small (i.e., not more than ε, a small positive value) if k is in some range, either m_1 or m_0 is larger than some threshold, and a few other conditions hold. If an upper bound on $E_{\mathcal{D},k}(m_1, m_0; \theta_1, \theta_0)$ becomes sufficiently small for all $k = 1, 2, \dots, n$, then their sum $E_{\mathcal{D}}(m_1, m_0; \theta_1, \theta_0) = \sum_k E_{\mathcal{D},k}(m_1, m_0; \theta_1, \theta_0)$ also becomes small; thus such thresholds on m_1 and m_0 respectively serve as upper bounds on the needed numbers of true and false examples, M_1^* and M_0^*.

Note that $E_{\mathcal{D},k}$ with "large" k or "small" k cannot be large for the following reason. Consider a pattern z with level k and an (m_1, m_0)-data set $X = X^1 \cup X^0$. If k is large (resp., small), then $|\mathbb{B}^n(z)| = 2^{n-k}$ tells that the z covers a small (resp., large) portion of binary vectors in $\mathbb{B}^n$. Thus the z is unlikely to be frequent in X^1 (resp., infrequent in X^0), and thus unlikely to be a (θ_1, θ_0)-pattern in X. Our analysis in the following is obtained by refining this observation.

Now let us introduce the following Chernoff's inequalities [24], well-known in the probability theory, whose variations can be found in [3], for example.

Theorem 5.4 (*Chernoff* [24]) *Assume that we are given a positive integer m and $\theta \in [0, 1]$.*

let Q_i be a random variable taking the value as follows:

$$Q_i = \begin{cases} 1-\theta & \text{with probability } \theta, \\ -\theta & \text{with probability } 1-\theta, \end{cases}$$

and let $Q = \sum_{i=1}^{m} Q_i$. Then, for any $\sigma > 1$,

$$\Pr(Q \geq (\sigma-1)\theta m) < (\exp(\sigma-1)\sigma^{-\sigma})^{\theta m} \tag{5.13}$$

holds.

Upper bounds on $E_{\mathcal{D},k}$ for a large k.

We derive two types of upper bounds on $E_{\mathcal{D},k}$ for "large" k by Theorems 5.5 and 5.6.

Theorem 5.5 *Assume that we are given a data source $\mathcal{D}$ satisfying Assumption 5.2, constants $\theta_1, \theta_0 \in [0,1]$ satisfying $\theta_1 > \theta_0$, a parameter $\varepsilon \in (0,1]$, m_1, m_0 and k. If $k \geq K_1$ and $m_1 \geq \overline{M_1^*}$, then $E_{\mathcal{D},k}(m_1, m_0; \theta_1, \theta_0) \leq \varepsilon$ holds, where*

$$K_1 = n - \log_2 \frac{\theta_1}{\exp(2)p_1}, \qquad \overline{M_1^*} = \frac{n\ln(2n) - \ln\varepsilon}{\theta_1}.$$

Proof: Let z be a pattern of level $k \geq K_1$. From Assumption 5.2 and $|\mathbb{B}^n(z)| = 2^{n-k}$, we have $\lambda_1^*(z;\mathcal{D}) \leq \min\{1, 2^{n-k}p_1\}$, and since $2^{n-k} \leq 2^{n-K_1} = \theta_1/(\exp(2)p_1)$, we have $\lambda_1^*(z;\mathcal{D}) \leq 2^{n-k}p_1 \leq \theta_1/\exp(2) < 1$. Let Q'_i be a random variable taking the value as follows:

$$Q'_i = \begin{cases} 1 & \text{with probability } 2^{n-k}p_1, \\ 0 & \text{with probability } 1-2^{n-k}p_1, \end{cases}$$

and let $Q' = \sum_{i=1}^{m_1} Q'_i$. We take another random variable Q_i defined by $Q_i = Q'_i - 2^{n-k}p_1$ and let $Q = \sum_{i=1}^{m_1} Q_i$ (and then $Q = Q' - 2^{n-k}p_1 m_1$). We have

$$\begin{aligned} E_{\mathcal{D},k}(m_1, m_0; \theta_1, \theta_0) &= \sum_{z \in Z_k} B_+(m_1, \theta_1, \lambda_1^*(z;\mathcal{D}))B_-(m_0, \theta_0, \lambda_0^*(z;\mathcal{D})) \\ &\leq B_+(m_1, \theta_1, 2^{n-k}p_1) \cdot |Z_k| \\ &= \Pr(Q' \geq \theta_1 m_1) \cdot 2^k \binom{n}{k} \\ &= \Pr\left(Q \geq 2^{n-k}p_1 m_1 \left(\frac{\theta_1}{2^{n-k}p_1} - 1\right)\right) \cdot 2^k \binom{n}{k}. \end{aligned}$$

From $k \geq K_1$, it holds $\theta_1/(2^{n-k}p_1) \geq \exp(2) > 1$. By applying Theorem 5.4 with $m = m_1$, $\theta = 2^{n-k}p_1$ and $\sigma = \theta_1/(2^{n-k}p_1)$, we have

$$\begin{aligned} E_{\mathcal{D},k}(m_1, m_0; \theta_1, \theta_0) &< \left(\frac{2^{n-k}p_1 \exp(1)}{\theta_1}\right)^{\theta_1 m_1} \cdot 2^k \binom{n}{k} \\ &\leq \left(\frac{2^{n-k}p_1 \exp(1)}{\theta_1}\right)^{\theta_1 m_1} \cdot (2n)^k \\ &\leq \exp(-\theta_1 m_1) \cdot (2n)^n. \end{aligned}$$

Then the inequality $\exp(-\theta_1 m_1) \cdot (2n)^n \leq \varepsilon$ is equivalent with $m_1 \geq \overline{M_1^*}$. $\square$

Another upper bound on $E_{\mathcal{D},k}$ for a large k is given by the following Theorem 5.6. It depends on a parameter ν_1 and can bound $E_{\mathcal{D},k}$ for k with $k > K_1 - 3$.

Theorem 5.6 *Assume that we are given a data source $\mathcal{D}$ satisfying Assumption 5.2, constants $\theta_1, \theta_0 \in [0,1]$ satisfying $\theta_1 > \theta_0$, a parameter $\varepsilon \in (0,1]$, m_1, m_0 and k. If $k \geq K_1(\nu_1)$ and $m_1 \geq \overline{M_1^*}(\nu_1)$ for some $\nu_1 \in (0, \theta_1)$, then $E_{\mathcal{D},k}(m_1, m_0; \theta_1, \theta_0) \leq \varepsilon$ holds, where*

$$K_1(\nu_1) = n - \log_2 \frac{\theta_1 - \nu_1}{p_1}, \qquad \overline{M_1^*}(\nu_1) = \frac{n \ln(2n) - \ln \varepsilon}{2\nu_1^2}.$$

Proof: For an arbitrary $\nu_1 \in (0, \theta_1)$, let z be a pattern of level $k \geq K_1(\nu_1)$. From Assumption 5.2 and $|\mathbb{B}^n(r)| = 2^{n-k}$, we have $\lambda_1^*(z; \mathcal{D}) \leq \min\{1, 2^{n-k}p_1\}$. Since $k \geq K_1(\nu_1)$, we have $2^{n-k}p_1 \leq \theta_1 - \nu_1 < \theta_1 \leq 1$. Thus $\lambda_1^*(z; \mathcal{D}) \leq 2^{n-k}p_1$ and

$$B_+(m_1, \theta_1, \lambda_1^*(z; \mathcal{D})) \leq B_+(m_1, \theta_1, 2^{n-k}p_1).$$

By applying (5.5) of Theorem 5.1 with $m = m_1$, $\theta = \theta_1$ and $p = 2^{n-k}p_1$, we have

$$B_+(m_1, \theta_1, 2^{n-k}p_1) \leq \exp(-2m_1(\theta_1 - 2^{n-k}p_1)^2),$$

and hence

$$E_{\mathcal{D},k}(m_1, m_0; \theta_1, \theta_0) \leq \exp(-2m_1(\theta_1 - 2^{n-k}p_1)^2) \cdot 2^k \binom{n}{k} \leq \exp(-2m_1\nu_1^2) \cdot (2n)^n.$$

Then the inequality $\exp(-2m_1\nu_1^2) \cdot (2n)^n \leq \varepsilon$ is equivalent with $m_1 \geq \overline{M_1^*}(\nu_1)$. $\square$

Note that the K_1 in Theorem 5.5 is a constant while $K_1(\nu_1)$ in Theorem 5.6 depends on the parameter ν_1. The following corollary is immediate from the definitions of K_1 and $K_1(\nu_1)$ and is useful in obtaining an upper bound $E_{\mathcal{D},k} \leq \varepsilon$ for such k with $K_1(\nu_1) \leq k \leq K_1$ from Theorem 5.6.

Corollary 5.2 *If we set $\nu_1 = \theta_1(1 - C/\exp(2))$ for a constant $1 \le C < \exp(2)$, then $K_1 - K_1(\nu_1) = \log_2 C$.*

Note that $K_1 - K_1(\nu_1) < \log_2 \exp(2) < 3$ holds.

An upper bound on $E_{\mathcal{D},k}$ for a small k.

Now we turn to an upper bound on $E_{\mathcal{D},k}$ for a "small" k.

Theorem 5.7 *Assume that we are given a data source $\mathcal{D}$ satisfying Assumption 5.2, constants $\theta_1, \theta_0 \in [0,1]$ satisfying $\theta_1 > \theta_0$, a parameter $\varepsilon \in (0,1]$, m_1, m_0 and k. If $k \le K_0(\nu_0)$ and $m_0 \ge \overline{M_0^*}(\nu_0)$ for some $\nu_0 \in (0,1)$, then $E_{\mathcal{D},k}(m_1, m_0; \theta_1, \theta_0) \le \varepsilon$ holds, where*

$$K_0(\nu_0) = n - \log_2 \frac{\theta_0 + \nu_0}{p_0}, \qquad \overline{M_0^*}(\nu_0) = \frac{K_0(\nu_0)\ln(2n) - \ln\varepsilon}{2\nu_0^2}.$$

Proof: The proof is similar to that of Theorem 5.6. For an arbitrary $\nu_0 \in (0,1)$, let z be a pattern of level $k \le K_0(\nu_0)$. From Assumption 5.2, $|\mathbb{B}^n(z)| = 2^{n-k}$ and $k \le K_0(\nu_0)$, we have $\lambda_0^*(z;\mathcal{D}) \ge 2^{n-k}p_0 \ge \theta_0 + \nu_0 > \theta_0$. By applying (5.6) of Theorem 5.1 with $m = m_0$, $\theta = \theta_0$ and $p = 2^{n-k}p_0$,

$$B_-(m_0, \theta_0, \lambda_0^*(z;\mathcal{D})) \le B_-(m_0, \theta_0, 2^{n-k}p_0) \le \exp(-2m_0(2^{n-k}p_0 - \theta_0)^2).$$

Hence we have

$$E_{\mathcal{D},k}(m_1, m_0; \theta_1, \theta_0) \le \exp(-2m_0(2^{n-k}p_0 - \theta_0)^2) \cdot 2^k \binom{n}{k} \le \exp(-2m_0\nu_0^2) \cdot (2n)^{K_0(\nu_0)}.$$

Then the inequality $\exp(-2m_1\nu_0^2) \cdot (2n)^{K_0(\nu_0)} \le \varepsilon$ is equivalent with $m_0 \ge \overline{M_0^*}(\nu_0)$. $\square$

An upper bound on $E_{\mathcal{D}}$.

Recall that Theorems 5.5 and 5.6 hold for a large k and Theorem 5.7 holds for a small k. Thus if one of these theorems holds for every $k = 1, 2, \ldots, n$, we have $E_{\mathcal{D}} = \sum_k E_{\mathcal{D},k} \le n\varepsilon$. More precisely, if we choose parameters ν_1 and ν_0 so that $K_1(\nu_1) \le K_0(\nu_0)$ holds, and we have $m_1 \ge \max\{\overline{M_1^*}, \overline{M_1^*}(\nu_1)\}$ and $m_0 \ge \overline{M_0^*}(\nu_0)$, then one of these theorems holds for every $k = 1, 2, \ldots, n$. A sufficient condition for $K_1(\nu_1) \le K_0(\nu_0)$ is given in the following corollary, immediate from the definitions of $K_1(\nu_1)$ and $K_0(\nu_0)$.

Corollary 5.3 *If $\nu_1 \in (0, \theta_1(1 - 1/\exp(2))]$ and $\nu_0 \in (0,1)$ satisfy $\nu_0 \le p_0(\theta_1 - \nu_1)/p_1 - \theta_0$, then $K_0(\nu_0) \ge K_1(\nu_1)$ holds.*

Finally, $E_{\mathcal{D}} = \sum_k E_{\mathcal{D},k}$ becomes sufficiently small under the conditions given in the following theorem.

Theorem 5.8 *Assume that we are given a data source $\mathcal{D}$ satisfying Assumption 5.2, constants $\theta_1, \theta_0 \in [0,1]$ satisfying $\theta_1 > \theta_0$, a parameter $\varepsilon \in (0,1]$, m_1 and m_0. If $\nu_1 \in (0, \theta_1(1-1/\exp(2))]$ and $\nu_0 \in (0,1)$ satisfy $\nu_0 \leq p_0(\theta_1 - \nu_1)/p_1 - \theta_0$, and $m_1 \geq \max\{\overline{M_1^*}, \overline{M_1^*}(\nu_1)\}$ and $m_0 \geq \overline{M_0^*}(\nu_0)$, then $E_{\mathcal{D}}(m_1, m_0; \theta_1, \theta_0) \leq n\varepsilon$ holds.*

Corollary 5.4 *For a random data source $\mathcal{D}_{\mathrm{rand}}$ (i.e., $p_1 = p_0 = 1/2^n$), there exist such $\nu_1 \in (0, \theta_1(1-1/\exp(2))]$ and $\nu_0 \in (0,1)$ that satisfy the condition $\nu_0 \leq \theta_1 - \nu_1 - \theta_0$ in Theorem 5.8. Then if we take ε sufficiently small (e.g., $\varepsilon = 2^{-n}$), $E_{\mathcal{D}}(m_1, m_0; \theta_1, \theta_0)$ converges to 0.*

Corollary 5.5 *The $\max\{\overline{M_1^*}, \overline{M_1^*}(\nu_1)\}$ and $\overline{M_0^*}(\nu_0)$ in Theorem 5.8 give upper bounds on the proposed necessary sizes M_1^* and M_0^*, respectively.*

Let us consider the possibility of using $\max\{\overline{M_1^*}, \overline{M_1^*}(\nu_1)\}$ and $\overline{M_0^*}(\nu_0)$ as estimates on M_1^* and M_0^*, respectively. To see how close they are, we computed $\overline{M_1^*}$, $\overline{M_1^*}(\nu_1)$ and $\overline{M_0^*}(s)$ on the random data source $\mathcal{D}_{\mathrm{rand}}$ for some combinations of (n, θ_1, θ_0), where we set ν_1 and ν_0 to the values that minimize $\max\{\overline{M_1^*}, \overline{M_1^*}(\nu_1), \overline{M_0^*}(\nu_0)\}$ among all $\nu_1 = C_1 \cdot 10^{-3} \in (0, \theta_1(1-1/\exp(2))]$ and $\nu_0 = C_0 \cdot 10^{-3} \in (0, \theta_1 - \theta_0 - \nu_1]$ with natural numbers C_1 and C_0. The obtained upper bounds, however, are not very tight; e.g., for $(n, \theta_1, \theta_0, \varepsilon) = (12, 0.10, 0.01, 1)$, $\overline{M_1^*} = 406.22$, $\overline{M_1^*}(\nu_1) = 6838.07$ and $\overline{M_0^*}(\nu_0) = 6817.67$. For UCI data sets with $n = 12$, i.e., BCW, CAR, HEART and TTT, however, $M_1^* + M_0^*$ is estimated as 255, 265, 250 and 245 (see Table 5.1), respectively.

The above discussion should indicate that the bounds $\max\{\overline{M_1^*}, \overline{M_1^*}(\nu_1)\}$ and $\overline{M_0^*}(\nu_0)$ are not very accurate indicators of M_1^* and M_0^*. It is left as our future work to derive tighter theoretical estimates of M_1^* and M_0^*.

5.5 Concluding remarks

In this chapter, we considered how many examples are sufficient or necessary in a given data set in order to remove deceptive (θ_1, θ_0)-patterns. In Section 5.3, we derived a sufficient data size by using Hoeffding's inequalities. In Section 5.4, for the derivation of a necessary data size, we claim that the data set should have at least a certain number of examples in the entire data space such that a random data set with the same number of examples contains (θ_1, θ_0)-patterns with a low probability. We gave some computational results to validate our claim. We also derived a theoretical upper bound on the proposed necessary data size as its estimate.

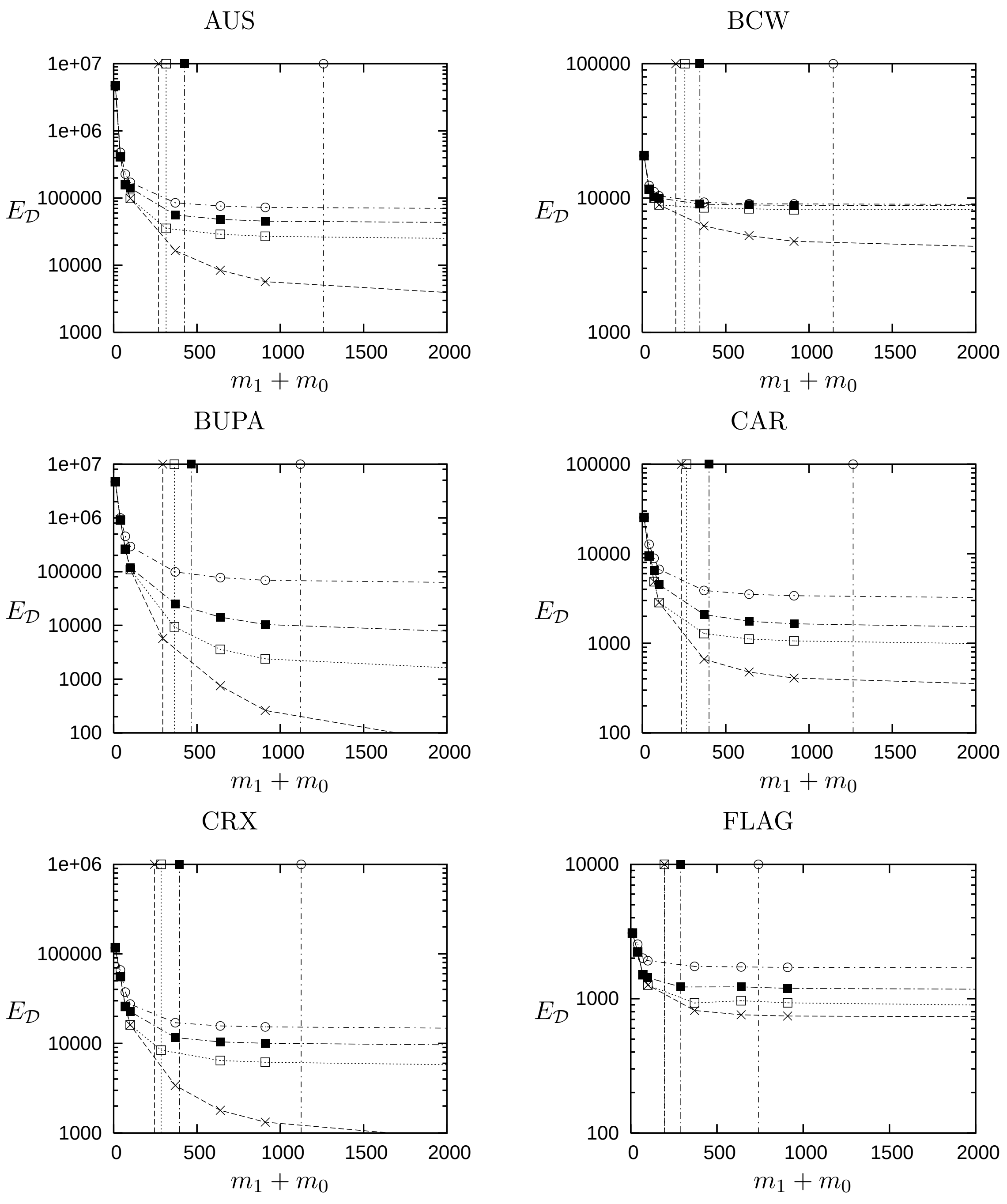


Figure 5.3: Expectations of the numbers of (θ_1, θ_0)-patterns in UCI data sets (AUS, BCW, BUPA, CAR, CRX and FLAG)

(Lines with points $\times, \square, \blacksquare, \circ$ represent $\theta_0 = 0.00, 0.01, 0.02, 0.05$, respectively. A broken line parallel to the vertical axis represents $M_1^* + M_0^*$.)

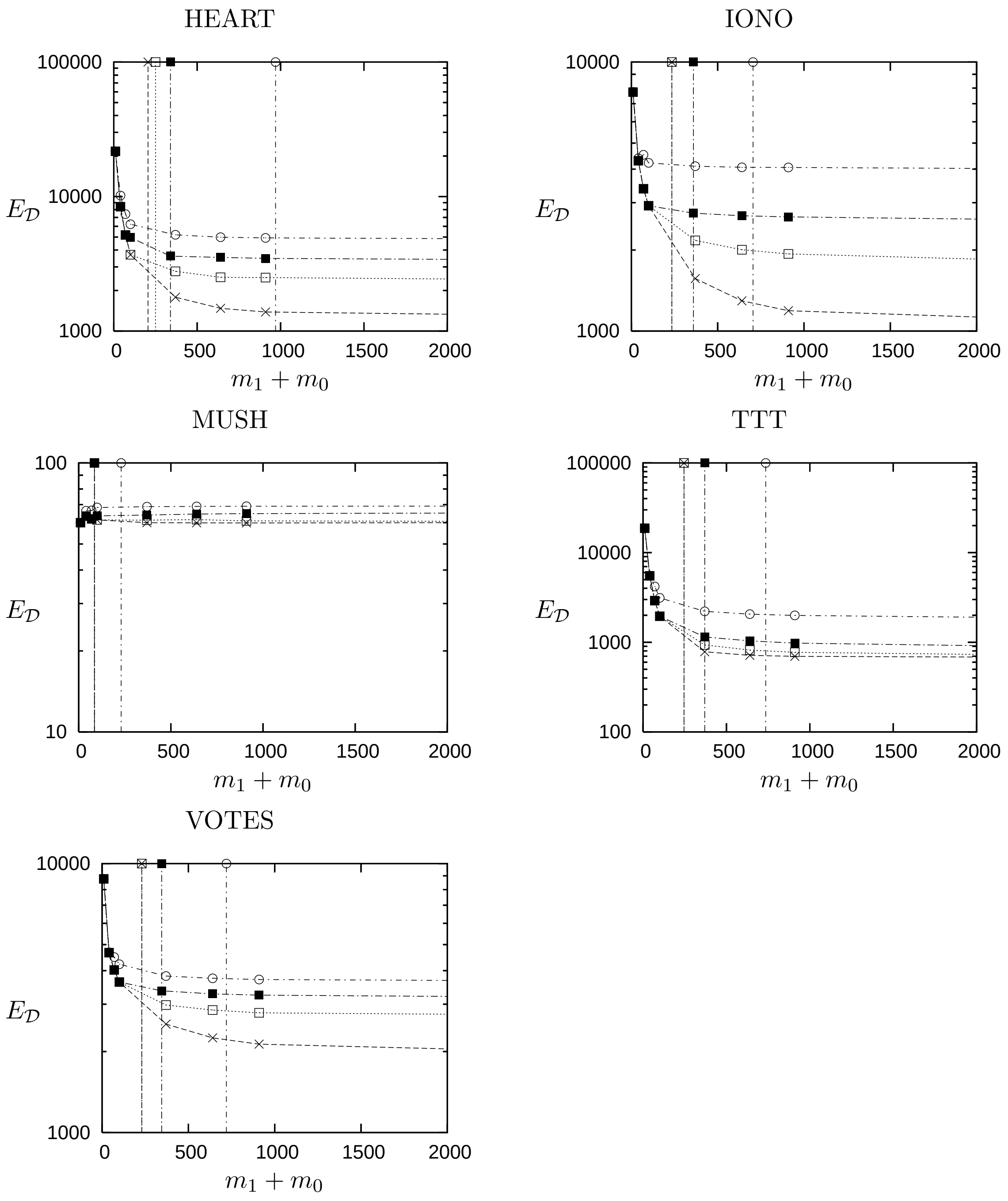


Figure 5.4: Expectations of the numbers of (θ_1, θ_0)-patterns in UCI data sets (HEART, IONO, MUSH, TTT and VOTES)

(Lines with points $\times, \square, \blacksquare, \circ$ represent $\theta_0 = 0.00, 0.01, 0.02, 0.05$, respectively. A broken line parallel to the vertical axis represents $M_1^* + M_0^*$.)

Chapter 6

Conclusion

Throughout this thesis, we considered the machine learning problem, mainly on the two-class supervised classification problem in step (L–3) and the data size used for learning in step (L–1) of Section 1.1.

In Chapter 2, we compared the potential performance between two representation models, ICF and DT. For this, we conducted theoretical and experimental analyses. In view of Occam's Razor, we showed the superiority of ICF to DT, as is summarized in Theorem 2.1. In Section 2.3, we conducted experimental studies on two feature determination schemes, majorization and extended majorization. The results showed that these schemes enable us to construct better ICF classifiers than decision trees, and that extended majorization can produce better features than majorization. Hence we consider that ICF is a promising model, and that it should be selected in step (C–1) of Section 1.1.1 in many cases.

In Chapter 3, we proposed two construction algorithms for ICF, ALG-ICF and ALG-ICF*. Both algorithms consist of composition process and selection process, but are significantly different in selection process. Also, ALG-ICF is based on features determined by majorization, while ALG-ICF* is based on those determined by extended majorization. Our computational experiments show that ALG-ICF* outperforms ALG-ICF both in true error rate and in computation time, and that ALG-ICF* is better than C4.5 for DT and BSVM for SVM in true error rate, provided that the parameter values are finely tuned up.

In Chapter 4, we considered extending ICF classifiers, originally proposed on $\mathbb{M}$-valued data sets, to real world data sets consisting of numerical and/or categorical attributes. In order to process such data sets by ICF, we first apply a discretization scheme to the given data set, and then construct a classifier from the discretized data set. We introduced two discretization schemes DC and SC, and based on the studies on their advantages and defects, we proposed a new one, IC. We observed that ALG-ICF$^*_{\mathrm{IC}}$ outperforms C4.5 in a stronger sense than the comparison in Chapter 3, i.e., ALG-ICF* constructs a better classifier than C4.5 by arbitrary parameter values. By this, we assert that ALG-ICF* is a new classifier construction algorithm in step (C–2) of Section 1.1.1, displaying a concept hierarchy on real world data sets.

In Chapter 5, we considered how many examples are sufficient or necessary in a given data set in order to remove deceptive (θ_1, θ_0)-patterns. In Section 5.3, we derived a sufficient data size by using Hoeffding's inequalities. In Section 5.4, for the derivation of a necessary data size, we claim that the data set should have a certain number of examples in the entire data space such that a random data set with the same number of examples contains (θ_1, θ_0)-patterns with a low probability. We gave some computational results to validate our claim, and then derived theoretical upper bounds on such numbers of examples.

The results from Chapters 2 to 4 show that ICF is substantially effective for UCI data sets, which are benchmark instances usually used for researcher and consist of at most 10^4 examples and 10^2 attributes, while many existing data sets are much larger. Our approach to such data sets may require exhausting computation time since the construction algorithm scans the whole data set many times. Hence we have to attain a scalable scheme for applications with large data sets. A possible solution to this is to construct an ICF classifier from an arbitrarily sampled subset of the whole data set. For this, the result of Chapter 5 may give a helpful suggestion: We analyzed a data size needed for removing deceptive (θ_1, θ_0)-patterns, which are simpler structure of knowledge than ICF classifiers. Then we may not expect to construct a good ICF classifier from the data set smaller than the derived necessary size. An extension of ICF to multi-class data sets is also a demanded task. These are left for our future work.

The author hopes that this thesis will provide some assistance to the research community of machine learning and artificial intelligence.

Bibliography

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD Conference*, pp. 207–216, 1993.

[2] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, Vol. 19, No. 6, pp. 716–723, 1974.

[3] N. Alon, J. H. Spencer, and P. Erdös. *The Probabilistic Method.* John Wiley & Sons, 1992.

[4] D. Angluin. Queries and concept learning. *Machine Learning*, Vol. 2, pp. 319–342, 1998.

[5] D. Angluin and P. Laird. Learning from noisy examples. *Machine Learning*, Vol. 2, pp. 343–370, 1988.

[6] M. Anthony. *Discrete Mathematics of Neural Networks - Selected Topics.* SIAM, 2001.

[7] M. Anthony and N. Biggs. *Computational Learning Theory.* Cambridge University Press, 1992.

[8] E. Arkin, H. Meijer, J. Mitchell, D. Rappaport, and S. Skiena. Decision trees for geometric objects. *International Journal of Computational Geometry and Applications*, Vol. 8, pp. 342–363, 1998.

[9] P. Auer, P. M. Long, and A. Srinivasan. Approximating hyper-rectangles: Learning and pseudorandom sets. *Journal of Computer and System Sciences*, Vol. 57, pp. 376–388, 1998.

[10] P. Auer and R. Ortner. A new PAC bound for intersection-closed concept classes. In *Learning Theory, 17th Annual Conference on Learning Theory* (*COLT 2004*), pp. 408–414, 2004.

[11] P. L. Bartlett. Lower bounds on the Vapnik-Chervonenkis dimension of multi-layer threshold networks. In *Proceedings of the 6th Annual ACM Conference on Computational Learning Theory* (*COLT 1993*), pp. 144–150, 1993.

[12] E. Baum and D. Haussler. What size net gives valid generalization? *Neural Computation*, Vol. 1, pp. 151–160, 1989.

[13] J. C. Bioch. The complexity of modular decomposition of boolean functions. *Discrete Applied Mathematics*, Vol. 149, pp. 1–13, 2005.

[14] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the Vapnik-Chervonekis dimension. *Journal of ACM*, Vol. 36, No. 4, pp. 929–965, 1989.

[15] A. Blumer, A. Enrenfeucht, D. Haussler, and M. K. Warmuth. Occam's razor. *Information Processing Letters*, Vol. 24, pp. 377–380, 1987.

[16] M. Bohanec and B. Zupan. A function-decomposition method for development of hierarchical multi-attribute decision models. *Decision Support Systems*, Vol. 36, No. 3, pp. 215–233, 2004.

[17] E. Boros, V. Gurvich, P. L. Hammer, T. Ibaraki, and A. Kogan. Decomposability of partially defined boolean function. *Discrete Applied Mathematics*, Vol. 62, pp. 51–75, 1995.

[18] E. Boros, V. Gurvich, L. Khachiyan, and K. Makino. On the complexity of generating maximal frequent and minimal infrequent sets. In H. Alt and A. Ferreira, editors, *Lecture Notes in Computer Science*, Vol. 2285, pp. 133–141. Springer-Verlag, 2002.

[19] E. Boros, P. L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, and I. Muchnik. An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No. 2, pp. 292–306, 2000.

[20] E. Boros, T. Ibaraki, and K. Makino. Logical analysis of binary data with missing bits. *Artificial Intelligence*, Vol. 107, pp. 219–263, 1999.

[21] E. Boros, T. Ibaraki, and K. Makino. Variations on extending partially defined boolean functions with missing bits. *Information and Computation*, Vol. 180, pp. 53–70, 2003.

[22] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, Vol. 2, No. 2, pp. 121–167, 1998.

[23] R. Butterworth, D. A. Simovici, G. S. Santos, and L. Ohno-Machado. A greedy algorithm for supervised discretization. *Journal of Biomedical Informatics*, Vol. 37, pp. 285–292, 2004.

[24] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, Vol. 23, pp. 493–509, 1952.

[25] C. K. Chow. On optimum recognition error and reject tradeoff. *IEEE Transacations on Information Theory*, Vol. 16, No. 1, pp. 41–46, 1970.

[26] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, Vol. 20, No. 3, pp. 273–297, 1995.

[27] Y. Crama, P. L. Hammer, and T. Ibaraki. Cause-effect relationships and partially defined boolean functions. *Annals of Operations Research*, Vol. 16, pp. 299–326, 1988.

[28] P. Domingos and M. J. Pazzani. Beyond independence: Conditions for the optimality of the simple bayesian classifier. In L. Saitta, editor, *Machine Learning, Proceedings of the 13th International Conference* (*ICML '96*), pp. 105–112, 1996.

[29] B. Efron and R. Tibshirani. Improvements on cross-valitation: The .632+ bootstrap method. *Journal of American Statistical Association*, Vol. 92, No. 438, pp. 548–560, 1997.

[30] A. Ehrenfeucht, D. Haussler, M. Kearns, and L. Valiant. A general lower bound on the number of examples needed for learning. *Information and Computation*, Vol. 82, pp. 247–261, 1989.

[31] T. Elomaa and J. Rousu. Fast minimum training error discretization. In C. Sammut and A. Hoffmann, editors, *Machine Learning, Proceedings of the 19th International Conference* (*ICML 2002*), pp. 131–138, 2002.

[32] U. M. Fayyad and K. B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, Vol. 8, pp. 87–102, 1992.

[33] U. M. Fayyad, G. Piatetsky-Shapiro, and S. Padhraic. From data mining to knowledge discovery in databases. *AI Magazine*, Vol. 17, No. 3, pp. 37–54, 1996.

[34] C. Frélicot and L. Mascarilla. Reject strategies driven combination of pattern classifiers. *Pattern Analysis and Applications*, Vol. 5, pp. 234–243, 2002.

[35] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, Vol. 55, pp. 119–139, 1997.

[36] J. H. Friedman. Recent advances in predictive (machine) learning. *Journal of Classification*, Vol. 23, pp. 175–197, 2006.

[37] T. Fulton, S. Kasif, and S. Salzberg. Efficient algorithms for finding multi-way splits for decision trees. In A. Prieditis and S. J. Russell, editors, *Machine Learning, Proceedings of the 12th International Conference on Machine Learning*, pp. 244–251, 1995.

[38] A. Gammerman, V. Vapnik, Y. LeCun, N. Bozanic, L. Bottou, C. Saunders, B. Schölkopf, A. Smola, M. O. Stitson, V. Vovk, C. Watkins, and J. A. E. Weston. *SVM software.* Royal Holloway and AT&T, http://svm.cs.rhul.ac.uk, 2001.

[39] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W. H. Freeman and Company, 1979.

[40] M. Garofalakis, D. Hyun, R. Rastogi, and K. Shim. Building decision trees with constraints. *Data Mining and Knowledge Discovery*, Vol. 7, pp. 187–214, 2003.

[41] P. Grünwal. Model selection based on minimum description length. *Journal of Mathematical Psychology*, Vol. 44, pp. 133–152, 2000.

[42] S. Hettich, C. L. Blake, and C. J. Merz. *UCI Repository of Machine Learning Databases.* Irvine, CA: University of California, Department of Information and Computer Science, http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998.

[43] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of American Statistical Association*, Vol. 58, pp. 13–30, 1963.

[44] P. G. Hoel, S. C. Port, and C. J. Stone. *Introduction to Statistical Theory.* Houghton Mifflin, 1971.

[45] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, Vol. 2, No. 5, pp. 359–366, 1989.

[46] C-W. Hsu and C-J Lin. A comparison on methods for multi-class support vector machines. *IEEE Transactions on Neuran Networks*, Vol. 13, pp. 415–425, 2002.

[47] C-W. Hsu and C-J Lin. A simple decomposition method for support vector machines. *Machine Learning*, Vol. 46, pp. 291–314, 2002.

[48] C-W. Hsu and C-J. Lin. *BSVM.* http://www.csie.ntu.edu.tw/~cjlin/bsvm/, 2006.

[49] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees in NP-complete. *Information Processing Letters*, Vol. 5, pp. 15–17, 1976.

[50] E. S. Laber and L. T. Nogueira. On the hardness of the minimum height decision tree problem. *Discrete Applied Mathematics*, Vol. 144, pp. 209–212, 2004.

[51] Y. Li and R. P. Gopalan. Effective sampling for mining association rules. In G. I. Webb and X. Yu, editors, *Lecture Notes in Computer Science*, Vol. 3339, pp. 391–401. Springer-Verlag, 2004.

[52] R. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, pp. 4–22, April 1987.

[53] W. Maass. Neural nets with superlinear VC-dimension. Technical Report TR94-017, The Electronic Colloquium on Computational Complexity (ECCC), 1994.

[54] O. L. Mangasarian. Data mining via support vector machines. In *System Modelling and Optimization*, pp. 91–112, 2001.

[55] S. Mii. Feature determination algorithms in the analysis of data. Master's thesis, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, March 2001.

[56] M. Minsky and S. Papert. *Perceptrons.* MIT Press, 1969.

[57] E. Mossel and C. Umans. On the complexity of approximating the VC dimension. *Journal of Computer and System Sciences*, Vol. 65, pp. 660–671, 2002.

[58] R. Muzzolini, Y. H. Yang, and R. Pierson. Classifier design with incomplete knowledge. *Pattern Recognition*, Vol. 31, No. 4, pp. 345–369, 1998.

[59] J. R. Noriega and H. Wang. A direct adaptive neural-network control for unknown nonlinearsystems and its application. *IEEE Transactions on Neural Networks*, Vol. 9, No. 1, pp. 27–34, 1998.

[60] H. Ono. *Structural Studies of Boolean Functions with Applications to Data Analysis.* PhD thesis, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, March 2002.

[61] H. Ono, K. Makino, and T. Ibaraki. Logical analysis of data with decomposable structures. *Theoretical Computer Science*, Vol. 289, pp. 977–995, 2002.

[62] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 130–136, 1997.

[63] C. H. Papadimitriou and M. Yannakakis. On limited nondeterminism and complexity of the V-C dimension. *Journal of Computer and System Sciences*, Vol. 53, pp. 161–170, 1996.

[64] J. Popova and J. C. Bioch. Monotone classification by function decomposition. In *Discovery Science*, pp. 203–214, 2005.

[65] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[66] J. R. Quinlan and R. L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, Vol. 80, pp. 227–248, 1989.

[67] B. D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.

[68] J. Rissanen. Modeling by the shortest data description. *Automatics*, Vol. 14, pp. 465–471, 1978.

[69] F. Rosenblatt. *Principles of neurodinamics: Perceptron and theory of brain mechanisms.* Spartan Books, 1962.

[70] D. Rumelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In D. Rumelhart and J. McClellanda, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* MIT Press, 1986.

[71] M. Schäfer. Deciding the Vapnik-Červonenkis dimension is $\sum_3^p$-complete. *Journal of Computer and System Sciences*, Vol. 58, pp. 177–182, 1999.

[72] B. Schölkopf and A. J. Smola. *Learning with Kernels.* MIT Press, 2002.

[73] J. C. Shafer, R. Agrawal, and M. Mehta. SPRINT: A scalable parallel classifier for data mining. In *Proceedings of the 22nd International Conference on Very Large Databases*, pp. 544–555, 1996.

[74] A. Shinohara. Complexity of computing Vapnik-Chervonenkis dimension and some generalized dimensions. *Theoretical Computer Science*, Vol. 137, pp. 129–144, 1995.

[75] H. Toivonen. Sampling large databases for association rules. In *Proceedings of the 22nd International Conference on Very Large Databases*, pp. 134–145, 1996.

[76] T. Uno, M. Kiyomi, and H. Arimura. LCM ver 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *Proccedings of IEEE International Conference on Data Mining '04* (*Workshop FIMI'04*), 2004.

[77] L. Valiant. A theory of learnable. *Communications of the ACM*, Vol. 27, pp. 1134–1142, 1984.

[78] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 2nd edition edition, 1999.

[79] V. N. Vapnik and A. Ja. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Doklady Akademii Nauk USSR*, Vol. 181, No. 4, 1968. In Russian.

[80] V. N. Vapnik and A. Ja. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, Vol. 16, pp. 264–280, 1971.

[81] 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 Kaufmann, 1991.

[82] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2nd edition, 2005.

[83] G. Yang. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 344–353, 2004.

[84] T. Young and T. Calvert. *Classification, Estimation, and Pattern Recognition.* Elsevier, 1973.

[85] B. Zupan, M. Bohanec, J. Demšar, and I. Bratko. Feature transformation by function decomposition. In H. Liu and H. Motoda, editors, *Feature Extraction, Construction and Selection*, chapter 20, pp. 325–340. Kluwer Academic Publishers, 1998.

A List of Author's Work

Journals

1. K. Haraguchi and T. Ibaraki. Construction of classifiers by iterative compositions of features with partial knowledge. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences,* Vol. E89-A, No. 5, pp. 1284–1291, 2006.

International Conferences with Review

1. K. Haraguchi and H. Nagamochi. Extension of ICF Classifiers to Real World Data Sets. In *Proceedings of the 20th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems* (*IEA/AIE 2007*), Kyoto, Japan, June 2007 (to be published).

2. K. Haraguchi and M. Yagiura. A randomness based analysis on the data size needed for generating reliable rules. In *Proceedings of the 2nd International Conference on Knowledge Engineering and Decision Support* (*ICKEDS 2006*), Lisbon, Portugal, May 2006, pp. 75–82.

3. K. Haraguchi, H. Nagamochi, and T. Ibaraki. Experimental studies on representation complexity and error rates of iteratively composed features. In *Proceedings of the 3rd International Conference on Innovative Applications of Information Technology for Developing World*, Kathmandu, Nepal, December 2005 (to be published).

4. K. Haraguchi, H. Nagamochi, and T. Ibaraki. Compactness of classifiers by iterative compositions of features. In *Proceedings of the 4th Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications*, Budapest, Hungary, June 2005, pp. 92–98.

5. K. Haraguchi, T. Ibaraki, and E.Boros. Classifiers based on iterative compositions of features. In *Proceedings of the 1st International Conference on Knowledge Engineering and Decision Support* (*ICKEDS 2004*), Porto, Portugal, July 2004, pp. 143–150.

6. K. Haraguchi, H. Ono, and T. Ibaraki. Construction of a rule-based classifier as optimization problems. In *Proceedings of International Conference on Optimization and Optimal Control*, UlanBator, Mongolia, August 2002, pp. 107.

Unpublished Manuscripts

1. K. Haraguchi, M. Yagiura, E. Boros, and T. Ibaraki. A randomness based analysis on the data size needed for removing deceptive patterns. submitted for publication.