社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－AL－111 (6)
2007／3／9

反復構成特徴に基づいた分類器の実データへの拡張

原口 和也　　永持 仁

京都大学大学院 情報学研究科 数理工学専攻 〒606-8501 京都市左京区吉田本町
E-mail: {kazuyah,nag}@amp.i.kyoto-u.ac.jp

摘要

分類問題とは, 与えられたデータ (オラクル関数によってラベルが付された事例の集合) を元に, オラクル関数に (近似的に) 等価な分類器を構成する問題である. 本論文では, 著者らが提案した $\{0,1,*\}$-値データ ($*$ は欠損値) に対する特徴反復構成アルゴリズム ALG-ICF* を, 数値や記号で記述されたより一般のデータに拡張する. ALG-ICF* による一般のデータの取扱いを可能にするため, 一般のデータを $\{0,1,*\}$-値データに変換する離散化スキームを考える. 従来の離散化スキームに対する考察を通じて離散化スキーム IC (integrated construction) を提案する. 前処理器として IC を備えた特徴反復構成アルゴリズム ALG-ICF$^*_{\mathrm{IC}}$ は, 決定木構成アルゴリズム C4.5 より優れた汎化能力を持つことを実験によって示した.

キーワード: 分類問題, データの離散化, 反復構成特徴, 機械学習

Extension of ICF Classifiers to Real World Data Sets

Kazuya Haraguchi　　Hiroshi Nagamochi

Department of Applied Mathematics and Physics,
Graduate School of Informatics, Kyoto University, Japan
E-mail: {kazuyah,nag}@amp.i.kyoto-u.ac.jp

Abstract

Classification problem asks to construct a classifier with good generalization from a given data set. Recently, we proposed an algorithm ALG-ICF* to construct a high performance classifier, which is based on iteratively composed features on $\{0,1,*\}$-valued data sets. In this paper, we extend ALG-ICF* so that it can also process real world data sets consisting of numerical and/or categorical attributes. We propose a new discretization scheme, integrated construction (IC), which transforms a real world data set into a $\{0,1,*\}$-valued one. The experiments reveal that ALG-ICF* with IC outperforms a decision tree constructor C4.5 in many cases.

Keywords: classification, discretization, iteratively composed features, machine learning

1 Introduction

Classification problem is one of the most significant issues in such fields as machine learning, artificial intelligence, logical analysis of data (LAD) [1], and so on, and is described as follows: We write the *data space* by $\mathbb{S}$. Let us denote the *oracle* by $y : \mathbb{S} \to \mathbb{B}$, where $\mathbb{B} = \{0,1\}$. (Thus there are two *classes*, 0 or 1.) The exact form of the oracle is not presented to us, but for some elements in $\mathbb{S}$, their classes are available. An *example* ω is an element of $\mathbb{S}$ whose class $y(\omega) \in \mathbb{B}$ is available. We call the set of examples the *data set*, denoted by Ω. Then classification problem asks us to find a function $c : \mathbb{S} \to \mathbb{B}$ that is an (approximately) equivalent function to y by utilizing the data set.

A typical approach to classification problem requires such a *representation model* that provides us with a framework of embodying a function by representing (or implementing) it as a structured object, called a *classifier*. There have been developed various kinds of representation models so far; e.g., linear discriminant functions, nearest neighbor classifiers, neural nets, decision trees, support vector machines [2].

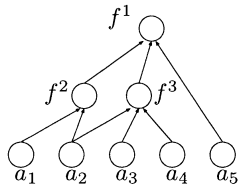

Figure 1: The structure of an ICF classifier

Recently, we proposed a new representation model ICF based on *iteratively composed features* [3]. ICF is established on an $\mathbb{M}$-valued data set (i.e., $\mathbb{S} = \mathbb{M}^n$), where $\mathbb{M} = \mathbb{B} \cup \{*\}$, $*$ denotes a *missing bit*, and n denotes the dimensionality. An *ICF classifier* $f : \mathbb{M}^n \to \mathbb{B}$ is based on functions called *features*: As a special case of feature, we introduce an *initial feature* $a_j : \mathbb{M}^n \to \mathbb{M}$ for each attribute $j = 1, 2, \ldots, n$, which is defined as $a_j(x) = x_j$ for $\forall x \in \mathbb{M}^n$. Then a feature f_S in general is a function $f_S : \mathbb{M}^S \to \mathbb{M}$ of the set S of other features. By transforming the range $\mathbb{M}$ into $\mathbb{B}$, a feature can be used as an ICF classifier. Figure 1 illustrates such an ICF classifier $f^1 = f_{\{f^2, f^3, a_5\}}$ with $f^2 = f_{\{a_1, a_2\}}$ and $f^3 = f_{\{a_2, a_3, a_4\}}$. An ICF classifier has a hierarchical structure of compositions from other features, and hence ICF is regarded as a generalization of *concept hierarchy* [4] or *decomposable Boolean functions* [5].

In our previous work [3], we proposed an algorithm ALG-ICF* for constructing an ICF classifier and observed from computational experiments that it *can* construct a better classifier (in the sense of generalization) than a decision tree constructor C4.5 [6] or support vector machines when its parameter values are finely tuned up. Hence not only is ICF an efficient representation model but also does it display interesting knowledge representation.

However, ICF works only on an $\mathbb{M}$-valued data set. In this paper, we extend ICF so that it can also handle real world data sets consisting of numerical and/or categorical attributes. We consider a date set Ω over an N-dimensional data space $\mathbb{S} = \mathbb{D}_1 \times \mathbb{D}_2 \times \cdots \times \mathbb{D}_N$, where the *domain* $\mathbb{D}_q$ of each attribute $q = 1, 2, \ldots, N$ is either numerical or categorical.

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 paper is to establish such a discretization scheme. Our discretization scheme constructs a set $D = \{\chi_1, \chi_2, \ldots, \chi_n\}$ of *discretizers*, where each discretizer $\chi_j \in D$ $(j = 1, 2, \ldots, n)$ is a mapping from $\mathbb{S}$ to $\mathbb{M}$, and 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. We first introduce two schemes arising from previous studies, called *domain based construction* (DC) and *space based construction* (SC): In DC, we regard a discretizer, a mapping from $\mathbb{S}$ to $\mathbb{M}$, 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 $q = 1, 2, \ldots, N$, 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 [7], and by the Naïve-Bayesian approach for a categorical attribute [8]. 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.

The paper is organized as follows. In Sect. 2, we first review the algorithm ALG-ICF*. In Sect. 3, with detailed description on a discretizer, we introduce discretization schemes DC and SC. Then in Sect. 4, we carry out experimental studies on ALG-ICF$^*_{\mathrm{DC}}$ and ALG-ICF$^*_{\mathrm{SC}}$. After studying the advantages and defects of both algorithms, we propose a new discretization scheme called *integrated construction* (IC), which is obtained by integrating DC and SC in an attempt to enhance the performance. Our experimental results show that ALG-ICF$^*_{\mathrm{IC}}$ performs effectively for real world data sets, and that it outperforms C4.5 in more cases than our previous work [3]. In Sect. 5, we make concluding remarks.

2 Algorithm ALG-ICF* on $\mathbb{M}$-valued data sets

This section reviews algorithm ALG-ICF* proposed to construct an ICF classifier on an $\mathbb{M}$-valued data set X [3]. First, let us describe how to determine function $f_S : \mathbb{M}^S \to \mathbb{M}$ for an $\mathbb{M}$-valued given set S of features. For a data set X, we call an example $x \in X$ labeled as $y(x) = 1$ (resp., 0) a *true* (resp., *false*) example. Let us denote by $X^1 = \{x \in X \mid y(x) = 1\}$ (resp., $X^0 = \{x \in X \mid y(x) = 0\}$) the set of true (resp., false) examples. For a vector $s \in \mathbb{M}^S$, let us denote by $X_{S,s}$ the set of examples in X whose projections on S are s. Let $X^1_{S,s} = X^1 \cap X_{S,s}$ and $X^0_{S,s} = X^0 \cap X_{S,s}$. We write the $f_S(x|_S)$ as $f_S(x)$ for convenience.

The output $f_S(s)$ is determined based on the following statistical test: The hypothesis is that true and false examples in $X_{S,s}$ are generated with the same probability. If the hypothesis is accepted, we let $f_S(s) = *$ (i.e., we cannot see the bias of classes). Otherwise, we let $f_S(s) = 1$ 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., $*$).

Now we are ready to describe the algorithm ALG-ICF* as follows.

Algorithm ALG-ICF*

Input: An n-dimensional $\mathbb{M}$-valued data set X with n initial features $A = \{a_1, a_2, \ldots, a_n\}$ and parameters $\alpha \in [0, 1], \beta \in [0, 1]$ and $\mu \in [0, 0.5]$.

Output: An ICF classifier $f : \mathbb{M}^n \to \mathbb{B}$.

Step 1: Let $F_0 := A$ and $t := 1$.

Step 2: $h := 2$.

Step 2-1 (Construction) : Let $F := F_0 \cup F_1 \cup \cdots \cup F_{t-1}$. Construct a set $F_{t,h}$ of features by:

$$F_{t,h} = \begin{cases} \{f_S \mid S \subseteq F, S \cap F_{t-1} \neq \emptyset, |S| = h, f_{S\setminus\{g\}} \in F_{t,h-1} \text{ for } \exists g \in S\} & \text{if } h > 2, \\ \{f_S \mid S \subseteq F, S \cap F_{t-1} \neq \emptyset, |S| = h\} & \text{otherwise.} \end{cases}$$

Step 2-2 (Selection) : Let $F' := \emptyset$. For each $x \in X$, we select such a feature f_S that achieves $f_S(x) = y(x)$ and that attains the smallest function value $\varphi(f_S)$ of (1) (see below) among all features in $F \cup F_{t,2} \cup \cdots \cup F_{t,h}$, and let $F' := F' \cup \{f_S\}$. (Hence F' becomes the set of *selected* features.) By the obtained feature set F', update the feature sets as $F_1 := F_1 \cap F', \ldots, F_{t-1} := F_{t-1} \cap F'$ and $F_{t,2} := F_{t,2} \cap F', \ldots, F_{t,h} := F_{t,h} \cap F'$.

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

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

Step 4: Output some $f_S \in F_0 \cup F_1 \cup \cdots \cup F_{t-1}$ and halt. (The details will be described below.)

In the feature selection process in Step 2-2, a feature f_S is evaluated by the following cost function;

$$\varphi(f_S) = (E(f_S, X) + \mu U(f_S, X))(\frac{1}{\Delta(f_S)})^\beta, \tag{1}$$

where β and μ are adjustable parameters, and

$$E(f_S, X) = \frac{|\{x \in X \mid f_S(x) \neq y(x), f_S(x) \neq *\}|}{|X|}, \quad U(f_S, X) = \frac{|\{x \in X \mid f_S(x) = *\}|}{|X|},$$

$$\Delta(f_S) = \frac{|\{s \in \mathbb{B}^S \mid f_S(s) \in \mathbb{B}\}|}{2^{|S|}}.$$

$E(f_S, X)$ is the error rate of f_S on the data set X (i.e., *empirical error rate*), $U(f_S, X)$ is the rate of examples on which uncertain decisions are made, and $\Delta(f_S)$ is the rate of input vectors in $\mathbb{B}^S$ (not those in $\mathbb{M}^S$) for which f_S makes a decisive classification. By small E, U and large Δ, f_S should attain a small $\varphi(f_S)$ and thus be evaluated highly.

In Step 4, we obtain an ICF classifier as follows: We first transform each feature $f_S \in F$ into a function $\hat{f}_S : \mathbb{M}^S \to \mathbb{B}$ by setting $\hat{f}_S(s) = 1$ or 0 for each $s \in \mathbb{M}^S$, based on the major class in $X_{S,s}$, and then choose a feature $\hat{f}_S$ with the smallest empirical error rate $E(\hat{f}_S, X)$.

3 Discretization Schemes DC and SC

3.1 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.,

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

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 $k-1$ 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 (2).

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 partitioned subset $P_1, P_2, \ldots, P_k$. Then a discretizer $\chi = (q, \mathcal{P}, \ell)$ discretizes a data element $\omega \in \mathbb{S}$ into $\ell(\kappa)$, where $\omega_q \in P_\kappa$ holds $(\kappa = 1, 2, \ldots, k)$. For convenience, we write the mapped value by $\chi(\omega)$ instead of $\ell(\kappa)$. 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 based on q and $\mathcal{P}$ and how to select q and $\mathcal{P}$ in each scheme are described in the subsequent subsections.

3.2 Discretization Scheme DC

Based on previous discretization schemes [8, 9], 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 u, where K specifies the maximum cardinality of partitions for numerical attributes, and $u \in [0, 1]$ denotes the cost given to an assignment of $*$ of 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 $\Omega_{q,\mathcal{P},\kappa} = \{\omega \in \Omega \mid \omega_q \in P_\kappa\}$, $\kappa = 1, 2, \ldots, k$. 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 $\kappa = 1, 2, \ldots, \kappa$ 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 $u|\Omega_{q,\mathcal{P},\kappa}|$, where the parameter u is used as the relative cost of an uncertain decision to an erroneous one. Then we define the misclassification cost of a discretizer $\chi = (q, \mathcal{P}, \ell)$ as the sum of such costs over the k partitioned subsets $\Omega_{q,\mathcal{P},1}, \Omega_{q,\mathcal{P},2}, \ldots, \Omega_{q,\mathcal{P},k}$;

$$\Gamma_{\mathrm{DC}}(\chi) = \sum_{\kappa=1,2,\ldots,k:\ \ell(\kappa)\neq *} \min\{|\Omega^1_{q,\mathcal{P},\kappa}|, |\Omega^0_{q,\mathcal{P},\kappa}|\} + u \sum_{\kappa=1,2,\ldots,k:\ \ell(\kappa)=*} |\Omega_{q,\mathcal{P},\kappa}|. \tag{3}$$

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

$$\ell(\kappa) = \begin{cases} * & \text{if } u|\Omega_{q,\mathcal{P},\kappa}| \le \min\{|\Omega^1_{q,\mathcal{P},\kappa}|, |\Omega^0_{q,\mathcal{P},\kappa}|\}, \\ 1 & \text{if } u|\Omega_{q,\mathcal{P},\kappa}| > \min\{|\Omega^1_{q,\mathcal{P},\kappa}|, |\Omega^0_{q,\mathcal{P},\kappa}|\} \text{ and } |\Omega^1_{q,\mathcal{P},\kappa}| > |\Omega^0_{q,\mathcal{P},\kappa}|, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Note that same labels may be assigned to plural κ-s. In DC, we determine the label ℓ by (4) for given q and $\mathcal{P}$.

How to select a partition for each attribute. For each attribute $q = 1, 2, \ldots, N$, 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 $D = \{\chi_1, \chi_2, \ldots, \chi_N\}$ of N discretizers. Note that the search of a partition is independent between attributes. 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., $|\mathcal{P}| \leq K$) in order to save computation time. Then we obtain an optimum partition $\mathcal{P}$ by solving the corresponding dynamic programming [7, 9]. 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 [8].

3.3 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 $\Gamma_{\mathrm{SC}} \in \{\Gamma_{\mathrm{SC,ERR}}, \Gamma_{\mathrm{SC,PAIR}}\}$ specifies the cost function to evaluate a discretizer set in the greedy algorithm. $\Gamma_{\mathrm{SC,ERR}}$ and $\Gamma_{\mathrm{SC,PAIR}}$ are called *data space error* and *unseparated pairs*, respectively. The algorithm selects discretizers based on the specified cost function, where the greedy algorithm for $\Gamma_{\mathrm{SC,PAIR}}$ was first proposed by Mii [10]. The other parameter $\mathbb{V} \in \{\mathbb{B}, \mathbb{M}\}$ specifies the cardinality of a considered partition by $|\mathbb{V}|$ (i.e., 2 or 3) and the range of a label by $\mathbb{V}$, i.e., we use such a discretizer $\chi = (q, \mathcal{P}, \ell)$ that satisfies $|\mathcal{P}| = |\mathbb{V}|$ and $\ell : \{1, \ldots, |\mathbb{V}|\} \to \mathbb{V}$ hold.

How to determine a label. Let us take a partition $\mathcal{P} = \{P_1, \ldots, P_{|\mathbb{V}|}\}$ on attribute q. Different from DC, we assign any value of $\mathbb{V}$ to some output value $\ell(\kappa)$ ($\kappa = 1, \ldots, |\mathbb{V}|$) of a label ℓ in order to distinct partitioned subsets $\mathbb{D}_q = P_1 \cup \cdots \cup P_{|\mathbb{V}|}$. Under this concept, we determine the label $\ell : \{1, \ldots, |\mathbb{V}|\} \to \mathbb{V}$ 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 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 $D = \{\chi_1, \chi_2, \ldots, \chi_n\}$ 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 $\Gamma_{\mathrm{SC,ERR}}(D)$ and unseparated pairs $\Gamma_{\mathrm{SC,PAIR}}(D)$ to be;

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

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

Let Γ_{SC} represent the cost function of either $\Gamma_{\mathrm{SC,ERR}}$ or $\Gamma_{\mathrm{SC,PAIR}}$. Starting with the empty set $D = \emptyset$, our greedy algorithm iterates selecting such χ that minimizes $\Gamma_{\mathrm{SC}}(D \cup \{\chi\})$ among N candidates from the N attributes and updating $D := D \cup \{\chi\}$; if there is no χ that attains $\Gamma_{\mathrm{SC}}(D \cup \{\chi\}) < \Gamma_{\mathrm{SC}}(D)$, then the greedy algorithm halts and outputs the final $D = \{\chi_1, \chi_2, \ldots, \chi_n\}$.

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 although $\Gamma_{\mathrm{SC,ERR}}(D) = 0$ 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

the usage of $\Gamma_{\mathrm{SC}} = \Gamma_{\mathrm{SC,PAIR}}$ may construct a discretizer set having 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 $\Gamma_{\mathrm{SC,PAIR}}(D'') > 0$).

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 (whose details are omitted), and use it as the candidate from attribute q.

4 Experimental Studies

Preparation for experiments. For the experiments, we use data sets from UCI Repository of Machine Learning [11] as real world data sets. The summary is shown in the leftmost column of Table 1, where N_{num} (resp., N_{cat}) denotes the number of numerical (resp., categorical) attributes.

Let 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$^*_C$ as follows:

(1) We divide Ω into 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 by applying C to the training set Ω_{train}, by which we obtain the $\mathbb{M}$-valued training set $X_{\mathrm{train}} = D(\Omega_{\mathrm{train}})$.

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

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

Results on ALG-ICF$^*_{\mathrm{DC}}$ and ALG-ICF$^*_{\mathrm{SC}}$. Now we show the experimental results in Table 1, 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$^*_{\mathrm{DC}}$, we show only the result of $K = 3$ and $u = 0.3$ in the table, which is fairly better than all tested values of $K = 2, 3, \ldots, 6$ and $u = 0.1, 0.2, \ldots, 0.5$.

The indicated value in an upper (resp., a lower) entry denotes the best (resp., average) error rate in all combinations of algorithm parameter values: For ALG-ICF$^*_{\mathrm{DC}}$ and ALG-ICF$^*_{\mathrm{SC}}$, we take $\alpha \in \{0.01, 0.05, 0.1, 0.25, 0.5\}$, $\beta = 0.3$, and $\mu \in \{0.1, \ldots, 0.5\}$. For C4.5 [6], we use 1%, 5%, 10%, 25%, 50%, 75%, 100% as its *confidence level*, which is the parameter to adjust the size of a final decision tree and is considered as the most influential parameter. Note that we exploit an algorithm that constructs C4.5 decision trees directly from Ω_{train} (not from X_{train}) since this paper discusses construction algorithms on real world data sets. A bold face indicates the error rate smaller than C4.5. A sign "$\star$" indicates that the smallest value in each row.

At the bottom of the Table 1, we show the average of presented error rates for all data sets. The row BEST 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 represents the average of error rates observed in all data sets and in all tested parameter 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 our previous work [3], we considered classification problem on a $\mathbb{B}$-valued data set and showed that ALG-ICF* outperforms C4.5 in BEST. Thus we see that, with the discretization schemes DC or SC, ALG-ICF* still performs well on real world data sets. In the following, we consider how to improve the discretization schemes so that ICF algorithms have a better performance in AVG.

Table 1: Best (upper) and average (lower) error rates ($\times 10^2$) of classifiers.

Data ($\lvert\Omega\rvert, N_{\mathrm{num}}, N_{\mathrm{cat}}$)	ALG-ICF$^*_{\mathrm{DC}}$ $K=3$ $u=0.3$	ALG-ICF$^*_{\mathrm{SC}}$ $\Gamma_{\mathrm{SC}}=\Gamma_{\mathrm{SC,ERR}}$ $\mathbb{V}=\mathbb{B}$	ALG-ICF$^*_{\mathrm{SC}}$ $\Gamma_{\mathrm{SC}}=\Gamma_{\mathrm{SC,ERR}}$ $\mathbb{V}=\mathbb{M}$	ALG-ICF$^*_{\mathrm{SC}}$ $\Gamma_{\mathrm{SC}}=\Gamma_{\mathrm{SC,PAIR}}$ $\mathbb{V}=\mathbb{B}$	ALG-ICF$^*_{\mathrm{SC}}$ $\Gamma_{\mathrm{SC}}=\Gamma_{\mathrm{SC,PAIR}}$ $\mathbb{V}=\mathbb{M}$	ALG-ICF$^*_{\mathrm{IC}}$	C4.5
BCW	⋆**3.71**	**4.18**	**4.26**	**4.73**	**4.18**	**4.12**	5.00
(683,9,0)	**4.78**	**4.85**	**4.98**	**5.2**	⋆**4.62**	**4.63**	5.21
BUPA	**36.82**	**34.10**	**36.35**	**36.12**	**35.66**	⋆**33.81**	37.12
(345,6,0)	39.90	**36.42**	**37.48**	**37.81**	38.24	⋆**36.27**	38.18
HABER	27.21	27.14	27.61	⋆**25.91**	26.66	27.34	26.27
(294,3,0)	⋆**27.64**	**28.26**	29.31	29.10	29.27	**28.17**	28.40
IONO	**11.59**	**11.42**	12.44	13.86	13.63	⋆**10.85**	12.32
(351,34,0)	**12.50**	**12.32**	13.53	15.89	16.13	⋆**12.27**	12.64
PIMA	⋆**24.60**	26.17	27.42	26.64	26.45	26.04	25.52
(768,8,0)	⋆**25.75**	28.41	29.43	29.22	29.81	28.04	27.90
AUS	**15.42**	**15.21**	**15.68**	**15.42**	⋆**15.13**	⋆**15.13**	15.95
(690,6,8)	⋆**16.61**	17.70	18.72	18.45	18.15	**17.35**	17.48
CRX	**13.45**	**13.11**	**13.79**	**13.45**	**13.66**	⋆**12.87**	14.33
(666,6,9)	⋆**14.76**	**16.65**	17.32	17.15	17.13	**16.46**	16.74
FLAG	10.92	⋆**9.69**	12.47	**10.51**	12.88	**10.00**	10.51
(194,10,18)	**12.61**	⋆**11.01**	14.93	14.75	15.10	**12.30**	12.77
HEART	⋆**18.88**	**22.29**	25.77	**19.99**	25.55	**20.66**	24.45
(435,0,16)	⋆**20.42**	**25.27**	28.37	**23.06**	29.44	**22.89**	25.50
CAR	7.02	6.94	6.55	⋆**1.12**	**1.38**	⋆**1.12**	2.36
(1728,0,6)	7.28	8.17	7.76	2.87	⋆**2.68**	2.88	2.73
MUSH	⋆**0.00**	0.23	0.23	**0.01**	⋆**0.00**	⋆**0.00**	0.01
(8124,0,22)	0.02	0.23	0.23	0.01	0.01	⋆**0.00**	0.01
TTT	24.80	8.18	13.84	**5.26**	13.96	⋆**5.11**	8.08
(958,0,9)	26.58	12.11	17.51	9.01	17.34	9.17	⋆8.54
VOTES	4.35	4.49	4.49	4.77	4.26	4.40	⋆3.98
(435,0,16)	5.04	5.42	5.42	5.73	5.72	5.86	⋆4.66
BEST	15.29	**14.08**	15.45	**13.67**	14.87	⋆**13.18**	14.30
AVG	16.45	15.90	17.30	16.01	17.20	⋆**15.09**	15.44

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.

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

As to evaluation function, the usage of $\Gamma_{\mathrm{SC}}=\Gamma_{\mathrm{SC,PAIR}}$ 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 Sect. 3.3, the usage of $\Gamma_{\mathrm{SC}}=\Gamma_{\mathrm{SC,PAIR}}$ may construct a discretizer set containing more detailed information on Ω than $\Gamma_{\mathrm{SC,ERR}}$, which may explain the above phenomena.

Algorithm ALG-ICF$^*_{\mathrm{IC}}$. From the above observation, we introduce a new discretization scheme, integrated construction (IC). Let us denote by D_C a discretizer set constructed by discretization scheme C. Then we integrate DC and SC as follows:

> *If* $N_{\mathrm{num}}>0$, *then we use* $D_{\mathrm{IC}}=D_{\mathrm{DC}(3,0.3)}\cup D_{\mathrm{SC}(\Gamma_{\mathrm{SC,ERR}},\mathbb{B})}$ *as the discretizer set. Otherwise, we use* $D_{\mathrm{IC}}=D_{\mathrm{DC}(3,0.3)}\cup D_{\mathrm{SC}(\Gamma_{\mathrm{SC,PAIR}},\mathbb{B})}$.

As seen from the result in Table1, ALG-ICF$^*_{\mathrm{IC}}$ outperforms C4.5 not only in BEST but also in AVG. It indicates that ALG-ICF$^*_{\mathrm{IC}}$ is better than C4.5 in a stronger sense than ALG-ICF$^*_{\mathrm{DC}}$ and

ALG-ICF$^*_{\mathrm{SC}}$ are.

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. We do not observe that, however, integration of more discretization schemes always enhances ICF classifiers. For example, let us introduce another discretization scheme IC′, where we construct a discretizer set by $D_{\mathrm{IC}'} = D_{\mathrm{DC}(3,0.3)} \cup D_{\mathrm{SC}(\Gamma_{\mathrm{SC,ERR}},\mathbb{B})} \cup D_{\mathrm{SC}(\Gamma_{\mathrm{SC,PAIR}},\mathbb{B})}$. ALG-ICF$^*_{\mathrm{IC}'}$ achieves 13.19 in BEST and 15.35 in AVG, which is slightly worse than ALG-ICF$^*_{\mathrm{IC}}$. Also, an integration method of this type increases the size of a constructed discretizer set. It can increase the computation time of ALG-ICF*. Hence an arbitrary integration does not necessarily attain a good result.

5 Conclusion

In this paper, 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 handle 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$^*_{\mathrm{IC}}$ outperforms C4.5 in many cases.

References

[1] Boros, E., Hammer, P. L., Ibaraki, T., Kogan, A., Mayoraz, E. and Muchnik, I.: An Implementation of Logical Analysis of Data, *IEEE Trans. Knowledge and Data Engineering*, Vol. 12, No. 2, pp. 292–306 (2000).

[2] Weiss, S. M. and Kulikowski, C. A.: *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*, Morgan Kaufmann (1991).

[3] Haraguchi, K. and Ibaraki, T.: Construction of classifiers by iterative compositions of features with partial knowledge, *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E89-A, No. 5, pp. 1284–1291 (2006).

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

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

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

[7] Fulton, T., Kasif, S. and Salzberg, S.: Efficient Algorithms for Finding Multi-way Splits for Decision Trees, in Prieditis, A. and Russell, S. J. eds., *Machine Learning, Proc. 12th Int'l Conf. Machine Learning*, pp. 244–251 (1995).

[8] Domingos, P. and Pazzani, M. J.: Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier, in Saitta, L. ed., *Machine Learning, Proc. 13th Int'l Conf.* (*ICML '96*), pp. 105–112 (1996).

[9] Elomaa, T. and Rousu, J.: Fast Minimum Training Error Discretization, in Sammut, C. and Hoffmann, A. eds., *Machine Learning, Proc. 19th Int'l Conf.* (*ICML 2002*), pp. 131–138 (2002).

[10] Mii, S.: Feature Determination Algorithms in the Analysis of Data, Master's thesis, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University (2001).

[11] Hettich, S., Blake, C. L. and Merz, C. J.: *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).