PAPER *Special Section on Discrete Mathematics and Its Applications*

Construction of Classifiers by Iterative Compositions of Features with Partial Knowledge

Kazuya HARAGUCHI[†a)], ***Student Member and*** **Toshihide IBARAKI**[††b)], ***Fellow***

SUMMARY We consider the classification problem to construct a classifier $c : \{0,1\}^n \mapsto \{0,1\}$ from a given set of examples (training set), which (approximately) realizes the hidden oracle $y : \{0,1\}^n \mapsto \{0,1\}$ describing the phenomenon under consideration. For this problem, a number of approaches are already known in computational learning theory; e.g., decision trees, support vector machines (SVM), and iteratively composed features (ICF). The last one, ICF, was proposed in our previous work (Haraguchi et al., (2004)). A feature, composed of a nonempty subset S of other features (including the original data attributes), is a Boolean function $f_S : \{0,1\}^S \mapsto \{0,1\}$ and is constructed according to the proposed rule. The ICF algorithm iterates generation and selection processes of features, and finally adopts one of the generated features as the classifier, where the generation process may be considered as embodying the idea of boosting, since new features are generated from the available features. In this paper, we generalize a feature to an extended Boolean function $f_S : \{0,1,*\}^S \mapsto \{0,1,*\}$ to allow partial knowledge, where $*$ denotes the state of uncertainty. We then propose the algorithm ICF* to generate such generalized features. The selection process of ICF* is also different from that of ICF, in that features are selected so as to cover the entire training set. Our computational experiments indicate that ICF* is better than ICF in terms of both classification performance and computation time. Also, it is competitive with other representative learning algorithms such as decision trees and SVM.

key words: *classification, Boolean functions, partially defined Boolean functions, learning algorithms, iteratively composed features*

1. Introduction

We consider the *classification problem* of learning a hidden *oracle* from a given set of *examples*. Let us denote $\mathbf{B} = \{0,1\}$. In this paper, an oracle $y : \mathbf{B}^n \mapsto \mathbf{B}$ is considered as an unknown Boolean function and each example $x \in \mathbf{B}^n$ is labeled by the value $y(x) \in \mathbf{B}$. We call a set of given examples a *training set* and denote is as X. We decompose X as $X = X^1 \cup X^0$, where $X^1 = \{x \in X \mid y(x) = 1\}$ and $X^0 = \{x \in X \mid y(x) = 0\}$. An example x in X^1 and X^0 is called a *true* example and a *false* example, respectively. The classification problem is to construct a *classifier* $c : \mathbf{B}^n \mapsto \mathbf{B}$ (i.e., a Boolean function) such that c realizes y exactly or approximately on the training set X. For this problem, several approaches are known in computational learning theory, such as decision trees, neural networks, support vector machines, statistical discriminant functions, and so forth [10], [14]–[16].

In our earlier work [7], we proposed a learning algorithm called *iteratively composed features* (ICF). Let $A = \{a_1, a_2, \ldots, a_n\}$ denote the set of n *attributes* of data, and we denote $a_j(x) = x_j$ for $x \in \mathbf{B}^n$. ICF iteratively generates *features* from simple ones composed of a few attributes to more complex ones composed of already constructed features, and finally adopts one of the generated features as the classifier. The complexity for implementing a given Boolean function by features is discussed in [8].

Let us consider each attribute $a \in A$ as a special type of features. A feature is in general composed of a nonempty subset S of other features, and is defined as a Boolean function $f_S : \mathbf{B}^S \mapsto \mathbf{B}$. If S is a singleton $S = \{g\}$, then we define $f_S = g$. (Thus, $f_{\{a\}} = a$ for each $a \in A$.) Once constructed, f_S can be used as a variable of a new feature.

Figure 1 illustrates a feature $f = f_S$ with $S = \{f^1, f^2, a_5\}$, where f^1 and f^2 are features composed of some sets of attributes. In the figure, nodes represent features whose input variables are depicted by the incoming arcs. Thus the final feature f has a hierarchical structure of compositions from other features. Given an input vector $x \in \mathbf{B}^5$, its classification proceeds as follows: the values $x_1, \ldots, x_5$ are first given to the corresponding attribute nodes $a_1, \ldots, a_5$. Then, we repeat determining the output value $f_S(x|_S)$ of a feature f_S from the input vector $x|_S$, where $x|_S$ denotes the projection of x on S. Here we emphasize that S may contain not only original attributes but also constructed features. In the rest of the paper, we abbreviate $f_S(x|_S)$ to $f_S(x)$, if no confusion arises. Then, in this case, after $f^1(x)$ and $f^2(x)$ are determined (we already know $a_5(x) = x_5$), x is classified into $f(x) \in \mathbf{B}$. If $f(x) = y(x)$ holds, we say that *f covers x*.

Let $X_{S,v}$ denote the set of examples $x \in X$ such that $x|_S = v$ holds, and $X^1_{S,v}$ and $X^0_{S,v}$ denote the sets of true and

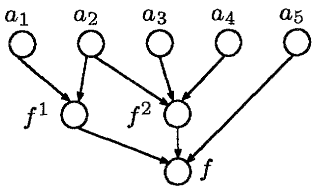


Fig. 1 A feature $f = f_{\{f^1, f^2, a_5\}}$.

Manuscript received August 20, 2005.
Manuscript revised November 9, 2005.
Final manuscript received December 22, 2005.

[†]The author is with the Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto-shi, 606-8501 Japan.
[††]The author is with the Department of Informatics, School of Science and Technology, Kwansei Gakuin University, Sanda-shi, 669-1337 Japan.
a) E-mail: kazuyah@amp.i.kyoto-u.ac.jp
b) E-mail: ibaraki@ksc.kwansei.ac.jp

DOI: 10.1093/ietfec/e89–a.5.1284

false examples in $X_{S,v}$, respectively. In ICF [7], given a training set X and a set S of features, f_S is determined by:

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

for each input vector $v \in \mathbf{B}^S$.

In this paper, we generalize the concept of a feature by allowing it to represent *partial knowledge*; we define a feature as $f_S : \mathbf{M}^S \mapsto \mathbf{M}$, where $\mathbf{M} = \{0, 1, *\}$ and $*$ denotes the state of uncertainty. To be more precise, we define $f_S(v) = 1$ (resp., 0) if $|X^1_{S,v}|$ is *decisively* larger (resp., smaller) than $|X^0_{S,v}|$, but define $f_S(v) = *$ otherwise. We distinguish the three cases by a certain statistical test, which is described in Sect. 2.

In Sect. 3, we propose an algorithm ICF* to generate such generalized features, and describe how to obtain a classifier $c : \mathbf{M}^n \mapsto \mathbf{B}$ from the generated features. Both ICF and ICF* consist of an iteration of the *generation process* and the *selection process*; in the former, new features are generated from the already generated features, and in the latter, some of them are selected to be maintained for the next iteration. Since they try to generate better features from the available features, the algorithms may be regarded as embodiments of the idea of boosting (e.g., [5]). In the last iteration, one feature from the maintained features is adopted as the classifier.

ICF and ICF* are different not only in feature types but also in selection processes. Let $E(f_S)$ denote the *error rate* of f_S on X;

$$E(f_S) = \frac{1}{|X|}\left(\sum_{f_S(v)=0} |X^1_{S,v}| + \sum_{f_S(v)=1} |X^0_{S,v}|\right). \tag{2}$$

ICF selects features in a greedy way on the basis of $E(f_S)$. ICF* selects features so that the selected features cover the entire training set well, as measured by the *classification cost*, which is also defined in Sect. 3.

The performance of the obtained classifier is evaluated by the error rate on the *test set* of examples. The test set is different from the training set, but is drawn from the same domain of the hidden oracle y. The error rate on the test set is defined by (2), but in this case, X is taken as the test set. We give some computational results in Sect. 4 to compare ICF* with ICF, a decision tree generator C4.5 [13], and a support vector machine (SVM) [6]. The results show that ICF* outperforms ICF, that ICF* can generate better classifiers than C4.5 in many cases, and that ICF* is competitive with SVM. We also observe that ICF* takes much less computation time than ICF. Finally, we give some concluding remarks in Sect. 5.

We note that the usage 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 [3], [4], [12], for example. For consistency, we assume in this paper that each example x may also contain $*$ as its attribute values. From a theoretical viewpoint of Boolean functions, functions $\mathbf{M}^n \mapsto \mathbf{M}$ have been studied intensively under the framework of partially defined Boolean functions and logical analysis of data (LAD) (e.g., [1], [2]).

2. Composition of Features

Let X denote a training set and S denote a nonempty subset of the available features (including A). We now consider how to define the corresponding feature $f_S : \mathbf{M}^S \mapsto \mathbf{M}$. For each $v \in \mathbf{M}^S$, the value $f_S(v)$ is defined so that it reflects the bias posed by the examples $X_{S,v}$. The bias is measured by a statistical test with significance level α, where α is a parameter specified by the user.

Assume that all examples are identically and independently distributed under some unknown distribution. We write the posterior probability of $y(x) = 1$ (resp., 0) under $x \in X_{S,v}$ as $p^1_{S,v}$ (resp., $p^0_{S,v}$), where $p^1_{S,v} + p^0_{S,v} = 1$ holds. It may be reasonable to determine $f_S(v) = 1$ (resp., 0) if $p^1_{S,v} > p^0_{S,v}$ (resp., $p^1_{S,v} < p^0_{S,v}$), and $f_S(v) = *$ if $p^1_{S,v} = p^0_{S,v} = 1/2$.

However, since the exact values of $p^1_{S,v}$ and $p^0_{S,v}$ are not known, we introduce the statistical test [9] with significance level α $(0 \leq \alpha \leq 1)$, in which we test whether or not the hypothesis $p^1_{S,v} = p^0_{S,v} = 1/2$ is valid from the given $|X^1_{S,v}|$ and $|X^0_{S,v}|$. If the hypothesis is rejected, then we conclude that there is a sufficient bias and we determine $f_S(v) = 0$ or 1 according to whether $|X^1_{S,v}| < |X^0_{S,v}|$ or $|X^1_{S,v}| > |X^0_{S,v}|$.

The statistical test is described as follows: Define $H(M, m)$ by:

$$H(M,m) = \left(\frac{1}{2}\right)^M \sum_{s=0}^{m} \binom{M}{s}, \tag{3}$$

where $M = |X^1_{S,v}| + |X^0_{S,v}|$ and $m = \min\{|X^1_{S,v}|, |X^0_{S,v}|\}$. If $H(M, m) \leq \alpha/2$, then we reject the hypothesis; otherwise, we accept the hypothesis. ($H(M, m)$ is the probability by which at most m true (or false) examples are generated in M examples under the hypothesis $p^1_{S,v} = p^0_{S,v} = 1/2$.)

Figure 2 shows the area of $(|X^1_{S,v}|, |X^0_{S,v}|)$ on which the hypothesis is rejected for $\alpha = 0.01, 0.1, 0.5, 1$. If the hypoth-

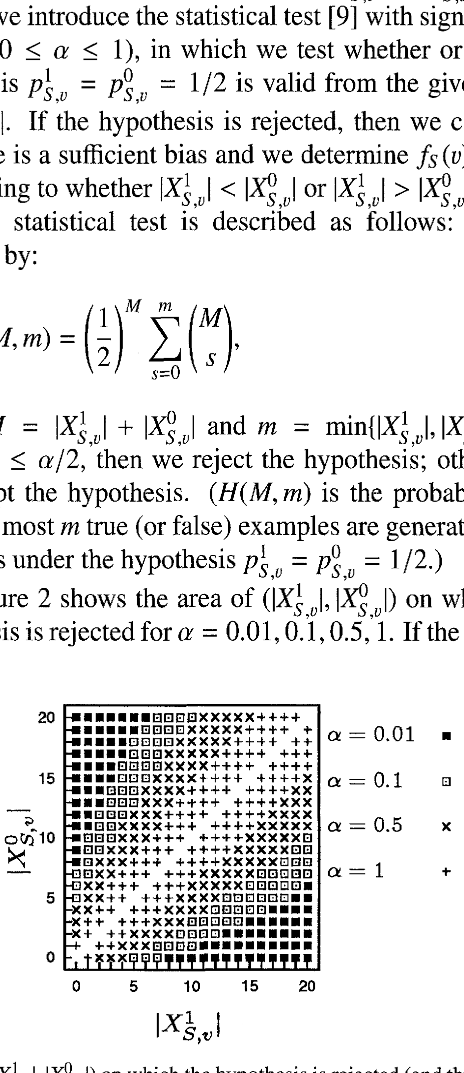


Fig. 2 $(|X^1_{S,v}|, |X^0_{S,v}|)$ on which the hypothesis is rejected (and thus $f_S(v) \in \mathbf{B}$).

esis on $(|X^1_{S,v}|, |X^0_{S,v}|)$ is rejected for an α_0, then it is also rejected for any $\alpha \geq \alpha_0$. Namely, for a larger value of α, $f_S(v)$ is more likely to be set to 0 or 1. Note that $H(M,m) > 1/2$ holds if and only if $|X^1_{S,v}| = |X^0_{S,v}|$ (i.e., $M = 2m$), and in this case, the hypothesis is always accepted for every value of $\alpha \in [0,1]$.

Now, given X, S and $\alpha \in [0,1]$, the value $f_S(v)$ for $v \in \mathbf{M}^S$ is determined by

$$f_S(v) = \begin{cases} * & \text{if } H(M,m) > \alpha/2, \\ 1 & \text{if } H(M,m) \leq \alpha/2 \text{ and } |X^1_{S,v}| > |X^0_{S,v}|, \\ 0 & \text{if } H(M,m) \leq \alpha/2 \text{ and } |X^1_{S,v}| < |X^0_{S,v}|. \end{cases} \tag{4}$$

Here we may have $|X^1_{S,v}| = |X^0_{S,v}| = 0$. In this case, $H(0,0) = 1 > \alpha/2$ for any $\alpha \in [0,1]$, and the rule (4) implies $f_S(v) = *$.

Since real examples are not always identically and independently distributed, the above scheme should be regarded as an approximation to real world data sets.

3. Algorithms ICF and ICF*

3.1 Common Framework of ICF and ICF*

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

Assume that we are in stage t. Let G_τ denote the set of features which are generated in stage τ ($0 \leq \tau < t$) and are maintained currently, where we let $G_0 = A$. Let $G := G_0 \cup \cdots \cup G_{t-1}$. In the $(d-1)$-st inner iteration ($d \geq 2$) of stage t, the generation process generates the set $G_{t,d}$ of features as follows;

$$G_{t,d} = \{f_S \mid S \subseteq G, S \cap G_{t-1} \neq \emptyset, |S| = d\}. \tag{5}$$

In other words, features $\{f_S\}$ are generated in the order of their sizes $|S| = d$.

The selection process then selects some features from $G \cup G_{t,2} \cup \cdots G_{t,d}$ by the *selection rule*, and prunes away the unselected features. The sets $G, G_{t,2}, \ldots, G_{t,d}$ are then reduced to the sets of such features that remain after the selection process. If the new set $G_{t,d}$ satisfies $G_{t,d} \neq \emptyset$, then the algorithms go to the next d-th inner iteration (and generate the set $G_{t,d+1}$ of features). On the other hand, if $G_{t,d} = \emptyset$ holds, we consider the following two cases: (i) $d = 2$ (i.e., no new feature is selected). The algorithms output the current set G of features and terminate. (ii) $d > 2$. The algorithms update $G := G \cup G_t$, where $G_t = G_{t,2} \cup \cdots \cup G_{t,d-1}$, and go to the next stage $(t+1)$.

The common description of the two algorithms is as follows.

Common Description of ICF and ICF*

```
Input: A training set $X$ with $n$ attributes $A = \{a_1, \ldots, a_n\}$
    and parameters (which are specified in each algorithm).
Output: A set $G$ of features.
Step 1: $G := G_0 := A$ and $t := 1$.
Step 2: $d := 2$.
    Step 2-1 (Generation) : Generate $G_{t,d}$ by (5).
    Step 2-2 (Selection) : Let $G'$ denote the features
        which are selected from $G \cup G_{t,2} \cup \cdots \cup G_{t,d}$
        by the selection rule. Update $G := G \cap G'$,
        $G_{t,2} := G_{t,2} \cap G', \ldots,$ and $G_{t,d} := G_{t,d} \cap G'$.
    Step 2-3: If $G_{t,d} \neq \emptyset$, then let $d := d + 1$ and return to
        Step 2-1.
Step 3: If $d > 2$, then let $G_t := G_{t,2} \cup \cdots \cup G_{t,d-1}$, $G :=$
    $G \cup G_t$, $t := t + 1$, and return to Step 2.
Step 4: Output $G$ and terminate.
```

ICF and ICF* are different in feature types and in selection rules. We review ICF in Sect. 3.2. In Sect. 3.3, we introduce the cost function for ICF*, called classification cost. In Sect. 3.4, we describe ICF* in more detail.

3.2 Algorithm ICF

In ICF, each $f_S \in G_{t,d}$ is treated as a Boolean function $f_S : \mathbf{B}^S \mapsto \mathbf{B}$, and is determined by (1).

Let $S = \{f^1, \ldots, f^d\}$ denote a set of d features, and let $S_j = S \setminus \{f^j\}$ $(j = 1, \ldots, d)$. Let $A(S)$ denote the set of attributes used to define the features in S (e.g., $A(S) = \{a_1, a_2, a_3, a_4, a_5\}$ for $f = f_S$ in Fig. 1). In the $(d-1)$-st inner iteration of stage t ($d \geq 2, t \geq 1$), the selection rule of ICF is described as follows;

```
Selection Rule (ICF): A set $G'$ of features is selected from
    the given sets $G \cup G_{t,2} \cup \cdots \cup G_{t,d}$ as follows.
Step 1. $G' := A$.
Step 2. For each feature $f_S \in (G \setminus A) \cup G_{t,2} \cup \cdots \cup G_{t,d}$,
    if $f_S$ satisfies all the following three conditions, then
    $G' := G' \cup \{f_S\}$.
(i) $f_{S_j} \in G \cup G_{t,2} \cup \cdots \cup G_{t,d}$ holds for all $j = 1, \ldots, |S|$.
(ii) $E(f_S) \leq \gamma E(f_{S_j})$ holds for all $j = 1, \ldots, |S|$, where $E$ is
    defined by (2) and $\gamma$ is a parameter $(0 \leq \gamma < 1)$.
(iii) $f_S$ has the smallest error rate on $X$ among the features
    in $G \cup G_{t,2} \cup \cdots \cup G_{t,d}$ having the same $A(S)$.
```

The following Prop. 1 [7] tells us that the error rate E on X is non-increasing as the generation proceeds. By (ii) of Step 2, we put a restrictive condition for a feature f_S to be selected.

Proposition 1: Let S and S^+ denote arbitrary sets of features such that $S \subset S^+$. Then, $E(f_S) \geq E(f_{S^+})$ holds for the features f_S and f_{S^+} constructed by (1).

Proof:

$$\begin{aligned} E(f_{S^+}) &= \frac{1}{|X|} \sum_{w \in \mathbf{B}^{S^+}} \min\{|X^1_{S^+,w}|, |X^0_{S^+,w}|\} \\ &= \frac{1}{|X|} \sum_{v \in \mathbf{B}^S} \sum_{u \in \mathbf{B}^{S^+ \setminus S}} \min\{|X^1_{S^+,(v,u)}|, |X^0_{S^+,(v,u)}|\} \\ &\leq \frac{1}{|X|} \sum_{v} \min\left\{\sum_{u} |X^1_{S^+,(v,u)}|, \sum_{u} |X^0_{S^+,(v,u)}|\right\} \end{aligned}$$

$$= \frac{1}{|X|}\sum_{v}\min\{|X^1_{S,v}|, |X^0_{S,v}|\}$$
$$= E(f_S).$$

□

The conditions (i) and (ii) say that a feature f_S of large S can be selected under a very tight condition. Due to the condition (iii), the number of maintained features is kept within 2^n (i.e., $|G| \leq 2^n$) during the execution of ICF.

Finally, from the output set G of features, the one $f_S : \mathbf{B}^S \mapsto \mathbf{B}$ attaining the smallest error rate on X is adopted as the classifier.

3.3 Classification Cost in ICF*

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

$$\varphi(f_S) = (E(f_S) + \mu U(f_S))\left(\frac{1}{D(f_S)}\right)^{\beta}, \tag{6}$$

where $E(f_S)$ is defined by (2), and

$$U(f_S) = \frac{1}{|X|}\sum_{f_S(v)=*}|X_{S,v}|, \tag{7}$$

$$D(f_S) = \frac{1}{2^{|S|}}|\{v \in \mathbf{B}^S \mid f_S(v) \in \mathbf{B}\}|. \tag{8}$$

Here $U(f_S)$ denotes the *uncertainty rate* of f_S on X, and $D(f_S)$ denotes the *decisiveness rate* as a function on the restricted domain $\mathbf{B}^S$ (whose size is $2^{|S|}$). μ and β are parameters to be set by the user, where μ is the cost given to a decision $f_S(v) = *$. In the computational experiments in Sect. 4, μ is set from 0.2 to 0.5, and β is set from 0.05 to 0.5.

It may appear that a definition $\varphi = E + \mu U$ (i.e., (6) with $\beta = 0$) is more natural for the cost function; in fact, it is used in the pattern recognition algorithms such as [3], [14]. 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 v whose components are all decisive (i.e., $v \in \mathbf{B}^S$). Based on this observation, we weight the cost $E + \mu U$ by $(1/D)^\beta$ with an appropriate parameter $\beta \geq 0$.

Given two sets $S \subset S^+$, we note that $v = w|_S$ and $w \in \mathbf{M}^{S^+}$ satisfy $|X_{S,v}| \geq |X_{S^+,w}|$, since $|X_{S,v}| = \sum_{w|_S=v}|X_{S^+,w}|$. Thus as a result of introducing the term $(1/D)^\beta$, we expect that $\varphi(f_S) \leq \varphi(f_{S^+})$ holds, since with a small $|X_{S^+,w}|$ $f_{S^+}(w) = *$ may hold for many $w \in \mathbf{B}^{S^+}$ under a relatively small α (e.g., $\alpha \leq 0.5$), as indicated in Fig. 2; it leads to a small $D(f_{S^+})$ and thus a large $\varphi(f_{S^+})$. In summary, the cost function (6) 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, the features selected by the selection process of ICF* are kept rather *robust* (i.e., not overfitting to the training set X). We discuss the influence of parameters α, β, μ later in Sect. 4.3.

3.4 Algorithm ICF*

In ICF*, each $f_S \in G_{t,d}$ is treated as $f_S : \mathbf{M}^S \mapsto \mathbf{M}$ by (4).

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

$$F(x) = \{f_S \in (G \setminus A) \cup G_{t,2} \cup \cdots \cup G_{t,d} \mid f_S(x) = y(x)\}, \tag{9}$$

i.e., $F(x)$ is the set of features (not including the attributes) covering x. The selection process of ICF* tries to maintain a set of features from $G \cup G_{t,2} \cup \cdots \cup G_{t,d}$ so that the resulting features, as partial knowledge, cover the entire training set X. 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 $\varphi(f_S)$ is selected. Therefore, an f_S not selected for any example $x \in X$ is pruned away. The selection process of ICF* is described as follows.

Selection Rule (ICF*): A set G' of features is selected from the given sets $G \cup G_{t,2} \cup \cdots \cup G_{t,d}$ as follows.

Step 1. $G' := A$.

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

Step 3. Using the resulting $G, G_{t,2}, \ldots, G_{t,d}$, construct $F(x)$ of (9) 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 $\varphi(f_S)$. If $f_S \notin G'$, then let $G' := G' \cup \{f_S\}$.

Note that the attributes A are not pruned by the selection process. It is due to the empirical reason that maintaining A makes the obtained features by ICF* better in terms of the error rates on the training set, and often on the test set.

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

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 (from Prop. 1) and U, $D \simeq 1$, and thus a small φ. Here, boosting (e.g., [5]) 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.

After generating a set G of features by ICF*, we modify each feature $f_S \in G$ to a function $f_S : \mathbf{M}^S \mapsto \mathbf{B}$ by determining $f_S(v) = 0$ or 1 by (1) for all $v \in \mathbf{M}^S$. The error rate $E(f_S)$ is then computed again, and we adopt the one attaining the smallest error rate on X as the classifier output by ICF*.

4. Computational Experiments

We construct classifiers by four approaches: ICF*, ICF, a decision tree generator C4.5 [13], and a support vector machine (SVM) [6]. As test instances, we use 10 artificial data sets and 10 real data sets. For each data set, we first define a training set and a test set. A classifier is then constructed from the former, and its performance is evaluated by the error rate on the latter. All computations are carried out on a PC (Pentium IV 2.8 GHz, memory 1 GB).

4.1 Data Sets

4.1.1 Artificial Data Sets

Each artificial data set has a linear threshold function as its oracle $y : \mathbf{B}^n \mapsto \mathbf{B}$ defined as follows; for each $x \in \mathbf{B}^n$,

$$y(x) = \begin{cases} 1 & \text{if } \sum_{j=1}^{n} w_j x_j \geq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

where we use $n = 14$, and we set each weight w_j at random ($|w_j| \leq 1$) for n_0 attributes ($n_0 \leq n$), and set $w_j = 0$ for the remaining $n - n_0$ attributes, in order to allow irrelevant attributes. Roughly speaking, it is easier to construct good classifiers on such a data set with a smaller n_0. We used $n_0 = 5, 6, \ldots, 14$, and call each data set `art5`, `art6`, ..., `art14`.

For given $\{w_j\}$, we generate 400 examples from $\mathbf{B}^n$ at random and use them as a training set, while we use the entire set $\mathbf{B}^n$ (i.e., 2^{14} examples) as the test set. For each $n_0 = 5, 6, \ldots, 14$, we generate 10 training sets and test sets in this way, and take the average of error rates on the 10 test sets.

4.1.2 Real Data Sets

We take 10 real data sets from UCI Machine Learning Repository [11] (i.e., `aus`, `bcw`, `bupa`, `car`, `crx`, `haber`, `heart`, `iono`, `pima`, `ttt`). The data sets contain from 250 to 1600 examples, and have from 3 to 34 numerical and/or categorical attributes. All data sets except `car` are two-labeled (i.e., $y(x) \in \mathbf{B}$); we modify `car` data set into a two-labeled data set. Also, some data sets contain contradicting examples and examples with missing values. Such examples are excluded in advance.

Each data set is then partitioned into halves at random, one for the training set and the other for the test set. We then binarize all examples by the method in [7] into binary examples, so that they are treated in our scheme, where the resulting binary data sets contain from 6 to 16 attributes. Each data set is partitioned 10 times, and we take the average of the error rates on the 10 test sets.

4.2 Parameter Values

All the four approaches have some program parameters, and they have more or less significant influence on the performance. In ICF*, we use all combinations of the parameters such that $\alpha \in \{0.005, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1\}$, $\beta \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$, and $\mu \in \{0.2, 0.3, 0.4, 0.5\}$. The influence of α, β, μ on ICF* is discussed in the next subsection.

ICF has a single parameter γ ($0 \leq \gamma < 1$), which controls the number of generated features; if γ is larger, then more features are generated and the computation time gets larger. We use $\gamma \in [0.75, 1.00)$ such that ICF halts within 3600 seconds.

In C4.5 [13], we use the *confidence level* of 0.1, 0.25, 0.5, 0.75, which is a parameter to adjust the target size of the final decision tree.

A nonlinear SVM is formulated in several ways [6], [10], [15]. The generator [6] adopted here uses a standard formulation (called a *generalized support vector machine* in [10]). In this formulation, the constructed classifier c is represented as follows;

$$c(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^{m} w_i v_i K(x, x^i) \geq \theta, \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

where $m = |X|$, $x^i \in X$ is the i-th example in X, $v_i = 2y(x^i) - 1$ (i.e., $v_i \in \{\pm 1\}$), $K : \mathbf{R}^n \times \mathbf{R}^n \mapsto \mathbf{R}$ is a *kernel function*, each w_i is determined by solving the following mathematical problem;

$$\begin{aligned} \max_{w_1, \ldots, w_m} \quad & \sum_{i=1}^{m} w_i - \frac{1}{2} \sum_{i,k=1}^{m} w_i w_k v_i v_k K(x^i, x^k) \\ \text{s.t.} \quad & \sum_{i=1}^{m} w_i v_i = 0, \\ & C \geq w_i \geq 0 \; (i = 1, \ldots, m), \end{aligned} \tag{12}$$

and θ in (11) is determined from the obtained $w_1, \ldots, w_m$. Note that C in (12) is the penalty for softening constraints in the primal problem. We use $C = +\infty$, implying that all classifiers attain no error on the training set. In this paper, we employ RBF Gaussian kernel functions,

$$K(x, x^i) = \exp(-r\|x - x^i\|^2), \tag{13}$$

and $r \in \{1/4, 1/2, 1, 2, 5\}$.

4.3 Results

4.3.1 Error Rate

We examine the best (i.e., smallest) average error rate of each approach, obtained from all possible parameter values. The results are shown in Table 1. We see that ICF* is better than ICF and C4.5 on almost all data sets, while it is competitive with SVM. We note that ICF* ranks first or second on all data sets. For the artificial data sets, it is reasonable that SVM exhibits a good performance, since it generates a classifier based on a hyperplane, while the oracle is also based on a hyperplane. We would rather emphasize that ICF* is much better than ICF and C4.5.

For `car` and `ttt`, the original data sets contain all possible examples. The training set, which is a half of the original data set, may contain sufficient information to construct

Table 1 Best error rates ($\times 10^2$) of ICF*, ICF, C4.5 and SVM.

	Data	ICF*	ICF	C4.5	SVM
	art5	**0.00**	9.29	**0.00**	0.01
	art6	**0.16**	8.74	0.63	0.26
	art7	**1.21**	14.12	3.03	1.39
	art8	**2.48**	13.50	4.71	2.79
artificial	art9	3.68	13.42	6.55	**3.52**
data sets	art10	5.77	15.39	10.00	**5.35**
	art11	6.12	16.86	10.66	**5.72**
	art12	8.28	19.08	13.96	**6.77**
	art13	9.97	25.08	17.99	**7.13**
	art14	10.77	20.09	19.65	**7.37**
	aus	15.19	**14.58**	15.57	18.66
	bcw	**4.09**	4.21	4.88	5.00
	bupa	34.45	35.78	**34.05**	37.80
	car	1.47	4.57	2.37	**0.72**
real	crx	**13.46**	13.67	14.95	18.62
data sets	haber	**26.53**	26.80	27.62	29.66
	heart	**21.78**	23.70	23.33	27.63
	iono	**15.11**	15.17	16.14	16.76
	pima	**26.25**	27.11	26.59	30.59
	ttt	5.76	17.10	11.09	**0.23**

Table 2 Computation time (sec.) of ICF*, ICF, C4.5 and SVM.

	Data	ICF*	ICF	C4.5	SVM
	art5	0.07	395.05	0.00	1.13
	art6	0.11	183.51	0.00	1.16
	art7	0.31	241.06	0.00	1.14
	art8	0.85	29.59	0.00	1.17
artificial	art9	1.81	76.08	0.00	1.17
data sets	art10	2.23	28.84	0.00	1.16
	art11	2.97	218.82	0.00	1.16
	art12	4.21	157.02	0.00	1.17
	art13	3.13	265.52	0.00	1.17
	art14	9.89	103.50	0.00	1.16
	aus	0.10	0.01	0.00	0.38
	bcw	0.15	90.94	0.00	0.14
	bupa	0.13	0.72	0.00	0.07
	car	4.33	694.90	0.01	11.59
real	crx	0.05	0.01	0.00	0.33
data sets	haber	0.01	0.18	0.00	0.05
	heart	0.11	0.02	0.00	0.04
	iono	0.08	0.09	0.00	0.07
	pima	2.14	38.11	0.01	0.44
	ttt	1.69	451.31	0.01	2.61

a good classifier, and thus a small error rate on the training set will mean a small error rate also on the test set. Since the parameter $C = +\infty$ for SVM results in a classifier with no error on the training set, SVM works very well on car and ttt; on the other hand, SVM appears to overfit to such data sets as aus and bupa.

For aus and bupa, we empirically know that a classifier with a simple structure is preferable; e.g., $f_{\{a\}} = a$ for an attribute $a \in A$ is better than almost all the reported classifiers on aus. ICF* generates more complex classifiers than ICF and C4.5, and thus its performance is worse (but is better than SVM).

4.3.2 Computation Time

Table 2 shows the average computation time of the four approaches. C4.5 outperforms the others in this respect. We note that ICF* takes much less computation time than ICF. In our experience, the computation time of ICF* and ICF is proportional to $|G|$, the number of maintained features, which is bounded as $|G| \leq n + |X|$ in ICF* and $|G| \leq 2^n$ in ICF. It may explain the difference in the computation time between ICF* and ICF. In fact, the sizes $|G|$ in ICF are much larger than those in ICF* in our computational results, though we omit the details.

4.3.3 Comparison of ICF* and ICF

As clear from the computational results, ICF* shows much better performance than ICF. Recall that ICF* and 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 ICF* classifiers have small depths in graph representations (like Fig. 1), and are achieved by small α from 0.01 to 0.1 (i.e., component features output $*$ for a nontrivial portion of inputs).

The effect of (ii) is also significant. ICF* maintains not only major features covering many examples but also minor features covering a small number of (exceptional) examples in the training set; ICF may maintain only major features due to its selection rule. We also conducted the computational comparison between ICF* and ICF with features $f_S : \mathbf{M}^S \mapsto \mathbf{M}$ by (4) (i.e., only the selection rules are different), and observed that the former gives better performance than the latter (the details are omitted).

4.3.4 Influence of Parameters

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

We first consider the influence of α. If α is large (e.g., $\alpha \geq 0.75$), then $f_S(v)$ and $f_{S^+}(w)$ are likely to be set to 0 or 1, and $D(f_S)$ and $D(f_{S^+})$ are close to 1 (hence so are $(1/D(f_S))^\beta$ and $(1/D(f_{S^+}))^\beta$ in (6)). Then, Prop. 1 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 $f_{S^+}(w)$ is likely to be set to $*$, whereas $f_S(v)$ is still likely to be set to 0 or 1 (note that $|X_{S,v}| \geq |X_{S^+,w}|$). In this case, $D(f_{S^+}) < D(f_S)$ implies $(1/D(f_S))^\beta < (1/D(f_{S^+}))^\beta$. Then, unless β is extremely small, an 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, note that $U(f_S) \leq U(f_{S^+})$ and $D(f_{S^+}) \leq D(f_S)$ usually hold. Then, if β and μ are large, we have $\varphi(f_{S^+}) > \varphi(f_S)$, and f_S composed of a small set S is preferred in the selection process, (thus the inner iteration tends to halt with a small d). If β and μ are small, the converse would be observed.

Figure 3 illustrates the above discussion about the influence of α, β on the size of d in one stage of ICF*, where a heavier color indicates a large d. Figure 4 gives the re-

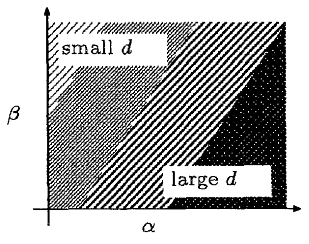


Fig. 3 Influence of α,β on the resulting d of the inner iterations.

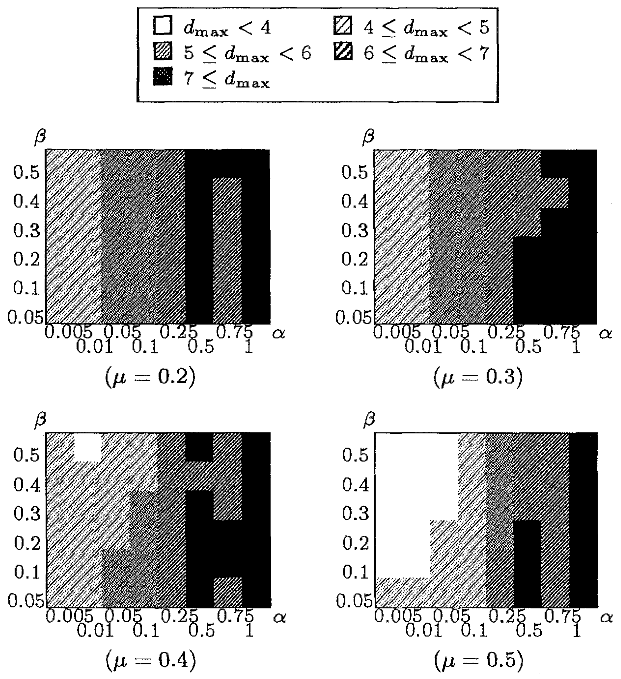


Fig. 4 $d_{\max}$ for various α,β,μ on art8.

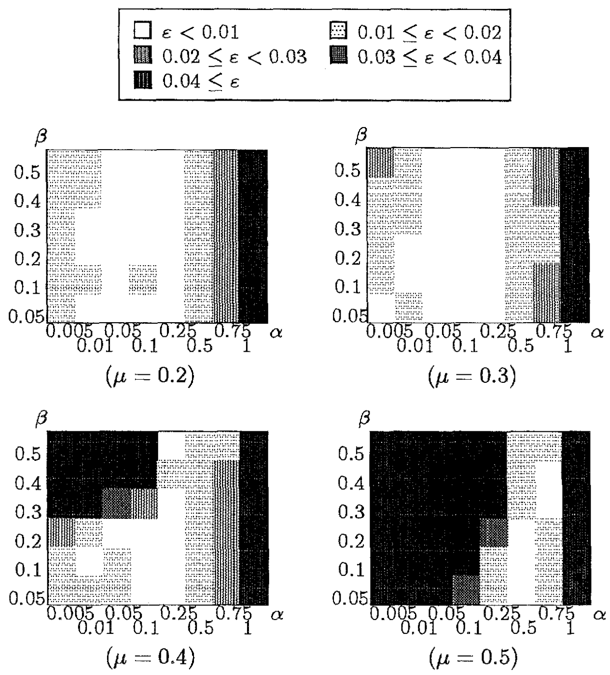


Fig. 5 Difference of error rates ε for various α,β,μ on art8.

sults observed on art8 data set. As the inner iteration may be executed more than once (i.e., in more than one stage) in each execution of 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 Fig. 4, for each μ, we show the average of $d_{\max}$ by colors (a heavier color means a large $d_{\max}$). We notice that the results show a tendency similar to that anticipated in Fig. 3.

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

To confirm the above observation, we consider the error rates realized by different values of α,β,μ (leading to different d) observed on art8 data set. For given α,β,μ, we denote the average error rate by $e(\alpha,\beta,\mu)$ and the smallest one by e^*, which is 2.48×10^{-2} on art8. We then define ε by

$$\varepsilon(\alpha,\beta,\mu) = e(\alpha,\beta,\mu) - e^*. \tag{14}$$

Figure 5 gives $\varepsilon(\alpha,\beta,\mu)$ on art8 for all α,β,μ. The tendency as discussed above is clearly shown here, since the middle areas attain rather small error rates, where $d_{\max}$ is from 5 to 7 approximately.

The results in Table 1 tell that ICF* can produce good classifiers if we are allowed to tune the parameters appropriately, which, however, may be difficult to be attained in practical situations. It is our future work to examine the criteria for determining appropriate parameter values.

5. Concluding Remarks

We proposed an improvement of the previous algorithm ICF [7] by introducing features with partial knowledge (i.e., with an output of $*$). The computational experiments indicate that the proposed algorithm ICF* can generate better classifiers. It is our future work to find an effective rule of determining appropriate parameter values. It is also desired to establish a theoretical foundation of ICF and ICF* approaches.

Acknowledgments

We gratefully acknowledge very careful and detailed comments given by anonymous reviewers. We are also indebted to Prof. H. Nagamochi of Kyoto University and to Prof. M. Yagiura of Nagoya University for a number of helpful suggestions. This work was partially supported by the Scientific Research on Priority Areas "New Horizon in Computing" by the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

[1] E. Boros, T. Ibaraki, and K. Makino, "Logical analysis of binary data with missing bits," Artif. Intell., vol.107, pp.219–263, 1999.
[2] E. Boros, T. Ibaraki, and K. Makino, "Variations on extending partially defined Boolean functions with missing bits," Inf. Comput., vol.180, pp.53–70, 2003.

[3] C.K. Chow, "On optimum recognition error and reject tradeoff," IEEE Trans. Inf. Theory, vol.16, no.1, pp.41–46, 1970.

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

[5] Y. Freund and R.E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," J. Comput. Syst. Sci., vol.55, pp.119–139, 1997.

[6] 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," [http://svm.cs.rhul.ac.uk], Royal Holloway and AT&T, 2001.

[7] K. Haraguchi, T. Ibaraki, and E. Boros, "Classifiers based on iterative compositions of features," Proc. 1st Intl. Conf. Knowledge Engineering and Decision Support, pp.143–150, Porto, Portugal, Aug. 2004.

[8] K. Haraguchi, H. Nagamochi, and T. Ibaraki, "Compactness of classifiers by iterative compositions of features," Proc. 4th Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications, pp.92–98, Budapest, Hungary, June 2005.

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

[10] O.L. Mangasarian, "Data mining via support vector machines," Proc. IFIP TC7 20th Conf. System Modeling and Optimization, pp.91–112, Trier, Germany, July 2001.

[11] P.M. Murphy and D.W. Aha, "UCI repository of machine learning databases," [http://www.ics.uci.edu/~mlearn/ MLRepository.html], 1992.

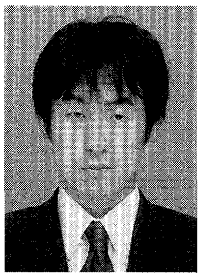
[12] R. Muzzolini, Y.H. Yang, and R. Pierson, "Classifier design with incomplete knowledge," Pattern Recognit., vol.31, no.4, pp.345–369, 1998.

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

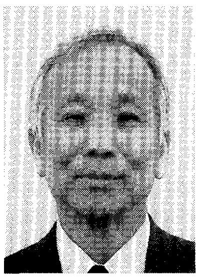
[14] B.D. Ripley, ed., Pattern Recognition and Neural Networks, Cambridge University Press, 1996.

[15] V. Vapnik, ed., The Nature of Statistical Learning Theory, Springer-Verlag, New York, 1995.

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



Kazuya Haraguchi received the B.E. in 2000 and the Master of Informatics from Kyoto University in 2002. He is currently in the doctoral program majoring in Applied Mathematics and Physics at the Graduate School of Informatics, Kyoto University. His interest includes algorithms, optimization, machine learning and their applications.



Toshihide Ibaraki received the B.E., M.E., and Ph.D. degrees in Engineering from Kyoto University, in 1963, 1965 and 1970, respectively. Since 1969, he had been with the Department of Applied Mathematics and Physics, Kyoto University, except for two and a half years from 1983 to 1985, during which time he was with Department of Computer and Information Sciences, Toyohashi University of Technology. After retiring from Kyoto University in March 2004, he is currently with Department of Informatics, School of Science and Technology, Kwansei Gakuin University. He has held a number of visiting appointments with University of Illinois, University of Waterloo, Simon Fraser University, Rutgers University, University of Canterbury and others. His interest includes algorithms, optimization, computational complexity and their applications.