Pergamon

Pattern Recognition, Vol. 29, No. 4, pp. 581–588, 1996
Elsevier Science Ltd
Copyright © 1996 Pattern Recognition Society
Printed in Great Britain. All rights reserved
0031-3203/96 $15.00 + .00

0031-3203(95)00107-7

CONSTRUCTION OF CLASS REGIONS BY A RANDOMIZED ALGORITHM: A RANDOMIZED SUBCLASS METHOD

MINEICHI KUDO, SHINICHI YANAGI and MASARU SHIMBO
Department of Information Engineering, Faculty of Engineering, Hokkaido University, Sapporo 060, Japan

(*Received* 23 *September* 1994; *in revised form* 5 *July* 1995; *received for publication* 18 *August* 1995)

Abstract—A randomized algorithm is proposed for solving the problem of finding hyper-rectangles, sufficiently approximating the true region in each class. This method yields a suboptimal solution, but is more efficient than previous methods. The performance is analysed based on a criterion of PAC (Probably Approximately Correct) learning. Experimental results show that the proposed method can solve large problems which were not able to be solved previously.

Subclass method Randomized algorithm Discrimination Hyper-rectangles
PAC learning Computational learning theory

1. INTRODUCTION

The problem of finding a region for each class in a feature space $\mathbf{R}^d$ is one of the most important issues in pattern recognition. Methods have been previously proposed for approximating the true region using hyper-rectangles.[1,2] Such hyper-rectangles have been used for both discrimination[1,2] and feature selection.[3,4] A method proposed by Kudo and Shimbo,[2,5] known as the *subclass method*, is a superior method for extracting structural information from classes and has been applied to feature selection.[4] Since the subclass method requires a combinational test of the training samples of a class, the time complexity is very large. Thus, pre- and postprocessings are usually used to reduce the number of training samples virtually. However, even if such processes are used, the following problems still remain: (1) to what degree the sample size should be reduced to and (2) to what degree does the reduction affect the resultant hyper-rectangles. If we use pre- and postprocessings, the answer will be suboptimal. The worst-case time complexity is still very large with respect to the number of reduced training samples and the actual time needed for a given problem is hard to estimate as the time depends strongly on the problem.

This paper proposes a randomized algorithm for suboptimally solving the problem described above and compares the algorithm with the subclass method (with pre- and postprocessings). With this algorithm, we can control the computation time by a parameter and solve the problem in a short time at the cost of the optimality of the solution. In this paper, we also analyse the algorithm's time complexity and evaluate the approximation performance using a criterion of PAC learning introduced by Valiant.[6] The experimental results show that the proposed algorithm is effective for large problems, which were not able to be solved by the subclass method without pre- and postprocessings.

2. SUBCLASS PROBLEMS

We consider a problem **SC-RECT** defined in reference (5). For a point set $X \subseteq \mathbf{R}^d$, let Rect(X) be the axis-parallel hyper-rectangle spanned by X:

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

where $\mathbf{x} = (x_1, x_2, \ldots, x_d) \in X$, $x_i \in \mathbf{R}$ $(i = 1, 2, \ldots, d)$.

Then, a problem **SC-RECT** is defined as follows:

Definition

A problem **SC-RECT**(n, m, d) is as follows:
INSTANCE: $S^+, S^- \subseteq \mathbf{R}^d$ $(|S^+| = n, |S^-| = m)$
ANSWER: $\Omega(S^+, S^-)$.

Here, $\Omega(S^+, S^-)$ is a subset family on S^+, of which element X is a maximal subset such that Rect(X) does not include any element of S^-.

We say that X is *exclusive* (*against* S^-) if Rect(X) does not include any $\mathbf{y} \in S^-$. We call $\Omega(S^+, S^-)$ a *subclass family* and each element $X \in \Omega(S^+, S^-)$ a *subclass*. The two input sets, S^+ and S^-, are called a *positive sample* set and a *negative sample set*, respectively. An example in **SC-RECT** is shown in Fig. 1.

The subclass method[2] finds all subclasses by a combinational examination of positive samples. Although the subclass method is originally implemented in a discrete domain, where each side of a hyper-rectangle is on one of the fixed thresholds, there is no essential

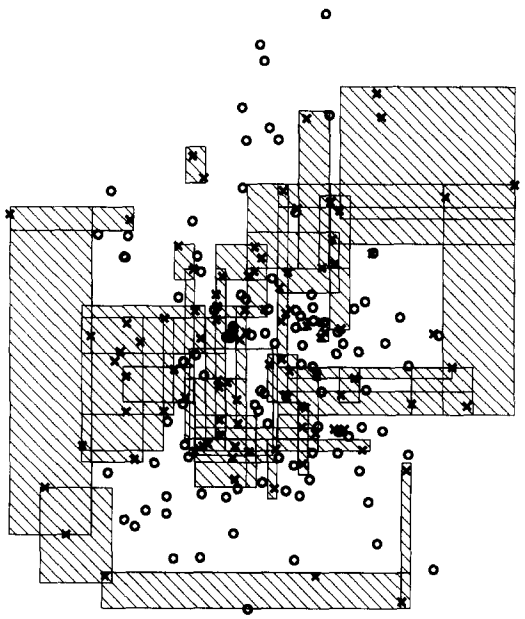


Fig. 1. An example of subclasses in **SC-RECT**. Positive samples are denoted by crosses and negative samples by open circles. A subclass is a set of positive samples which are included in a hatched rectangle.

difference between such a discrete formalization and this continuous formalization. The subclass method requires a large time complexity with respect to the number of positive samples. Therefore, the subclass method is usually carried out with the support of pre- and postprocessings. The preprocessing combines the nearest pair of samples of S^+ into one and then the next nearest pair into one and so on [for details, see reference (2)]. The combination of two positive samples $\mathbf{x}_1$ and $\mathbf{x}_2$ yields a hyper-rectangle $\mathrm{Rect}(\{\mathbf{x}_1, \mathbf{x}_2\})$ as a reduced positive sample. This process reduces the number of positive samples by about half in each round and is repeated until an appropriate number of reduced positive samples is obtained. After that, the ordinary subclass method is applied to the reduced positive samples and the original negative samples. Moreover, in the postprocessing, in order to make all obtained subclasses maximal, we add all the positive samples to every subclass unless their addition destroys its exclusiveness.

The subclass method with pre- and postprocessings gives a suboptimal solution and still requires a large time complexity with respect to the number of the reduced positive samples. Here, a solution is said to be *optimal* when it includes all possible subclasses and *suboptimal* when it includes only part of subclasses. The actual computation time depends strongly on the problem and is hard to estimate.

3. A RANDOMIZED ALGORITHM

In this section, we describe a randomized algorithm for obtaining subclasses.

We produce a subclass X from $S^+ = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ by the following basic procedure.

Basic procedure

Firstly, let X be empty. For a permutation $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$, examining $\mathbf{x}_{\sigma_1}, \mathbf{x}_{\sigma_2}, \ldots, \mathbf{x}_{\sigma_n}$ in the order, we add all these samples to X unless the addition damages the exclusiveness of X.

Then, the following proposition is satisfied.

Proposition 1. Let X_σ be a subset of S^+ obtained by the basic procedure for a permutation σ. Then, X_σ is a subclass.

Proof. The subset X_σ is obviously exclusive from the procedure. Let us show its maximality. Let us assume that X_σ is not maximal. Then there exists another exclusive subset X' such that $X_\sigma \subset X'$. When we pick an element $\mathbf{x}'$ from $X' - X_\sigma$, $X_\sigma \cup \{\mathbf{x}'\}$ is exclusive. Since $\mathbf{x}'$ must have appeared somewhere in the examination of $\mathbf{x}_{\sigma_1}, \mathbf{x}_{\sigma_2}, \ldots, \mathbf{x}_{\sigma_n}$, $\mathbf{x}'$ must have been added to X_σ. This is a contradiction. □

The proposition guarantees that we can always obtain a subclass in an examination of S^+ according to a permutation. There are $n!$ possible permutations. By repeating this simple procedure, we obtain a randomized algorithm. We term the algorithm a *randomized subclass method*, which is shown in Fig. 2. In

```
main
begin
      /* S+ : a positive sample set */
      /* S- : a negative sample set */
      /* s : the number of permutations to be examined */
      Ω ← φ;
      PSUB(S+,S-,Ω,s)
end;

procedure PSUB(S+,S-,Ω,s)
begin
(1)  n ← |S+|,   v ← 1;
(2)  for k ← 1 to s do
       begin
(3)      T ← S+;
(4)      X ← φ;
(5)      for i ← 0 to n − 1 do
           begin
(6)          r ← { uniform random number over [0,1)};
(7)          j ← ⌊r * (n − i)⌋ + 1;
(8)          x ← {the jth element of T};
(9)          T ← T − {x};
(10)         if X ∪ {x} holds exclusiveness against S- then
(11)             X ← X ∪ {x};
           end
(12)     if X does not exist in Ω then
           begin
(13)         Ω ← Ω ∪ {X};
(14)         v ← v − 1/(n choose |X|);
(15)         if v ≤ 0 then
(16)             break;
           end
       end
(17)   output Ω;
end;{PSUB}
```

Fig. 2. A randomized subclass method.

steps (5)–(11) of this algorithm, a permutation σ is chosen and a subclass X_σ is obtained. In step (12), duplicate registration of subclasses is avoided. Steps (14)–(16) give the termination condition. The rationale of this condition is given in the next section. It should be noted here that this condition holds soon if the first subclass X is S^+ itself, because $1/\binom{n}{|S^+|} = 1$.

The time complexity is $O(sn|\Omega|)$, which is consumed in step (12), where s is a predetermined number of iterations, i.e. the number of permutations to be examined. How large a value of s do we need to attain a good approximation for the optimal solution Ω? To answer this question, we use an evaluation proposed in the PAC learning paradigm, as discussed in the next section.

Stoffel[7] began such a trial to find subclasses, which he calls *prime events*. He proposed a similar procedure for examining positive samples in the presented order. Therefore, we can regard our algorithm as a randomized variation of Stoffel's algorithm. His algorithm, however, repeats this procedure only for samples which are not used in previous procedures. Thus, basically his subclasses have no common elements, which make them different from ours.

Our randomized algorithm has the following two merits: (1) we can obtain a suboptimal solution (part of all possible subclasses) whenever we wish by adjusting the iteration parameters s and (2) we can find almost all the important subclasses in the early stages. The first merit above is very important if we want to deal with large problems and the second merit will be discussed in detail later.

4. PERFORMANCE OF THE RANDOMIZED ALGORITHM

In this section, we discuss the performance of our randomized algorithm. First, we estimate an upper bound of $|\Omega|$.

4.1. *Sizes of subclass families*

Let Φ be a subset family on a set S of n elements. We call Φ a *Sperner family* when no one element $X \in \Phi$ contains another $X' \in \Phi$. By the definition, a subclass family Ω is obviously a Sperner family on S^+.

Then, the following theorem holds:[8,9]

Theorem 1. (Sperner, Lubell). Let Φ be a Sperner family on $S(|S| = n)$. Then:

$$|\Phi| \leq \binom{n}{\lfloor n/2 \rfloor},$$

and more strongly

$$\sum_{X \in \Phi} \frac{1}{\binom{n}{|X|}} \leq 1.$$

From these, the following is immediately derived.

Corollary 1. For every element X of a Sperner family Φ on S and any $d \leq \lfloor n/2 \rfloor$, if $|X|$ is not beyond d or not under $(n-d)$, then:

$$|\Phi| \leq \binom{n}{d}.$$

Proof. Since $\binom{n}{i} \leq \binom{n}{d}$ for i such that $i \leq d$ or $i \geq n-d$, by Theorem 1:

$$\sum_{X \in \Phi} \frac{1}{\binom{n}{|X|}} \leq 1$$

$$\sum_{X \in \Phi} \frac{1}{\binom{n}{d}} \leq \sum_{X \in \Phi} \frac{1}{\binom{n}{|X|}} \leq 1$$

$$|\Phi| \frac{1}{\binom{n}{d}} \leq 1$$

$$|\Phi| \leq \binom{n}{d}.$$

□

This follows Theorem 2.

Theorem 2. Let (S^+, S^-) be an instance of a subclass problem **SC-RECT**(n, m, d) and $\Omega(S^+, S^-)$ be the solution. Then:

$$|\Omega| \leq \binom{n}{\min\{2d, \lfloor n/2 \rfloor\}}.$$

Proof. Since $|\Omega| \leq \binom{n}{\lfloor n/2 \rfloor}$ is clear by Theorem 1, we prove $|\Omega| \leq \binom{n}{2d}$. Let us consider a subclass X and the corresponding hyper-rectangle Rect(X). Then, Rect(X) includes at most $2d$ points that determine two edges of Rect(X) in each axis. Let $\mathbf{m}^i \in X$ be the element that has the minimum value in the ith axis. In a similar way, we use $\mathbf{M}^i$ to show the element that has the maximum value in the ith axis. Then a set $X_{2d} = \{\mathbf{m}^1, \mathbf{m}^2, \ldots, \mathbf{m}^d, \mathbf{M}^1, \mathbf{M}^2, \ldots, \mathbf{M}^d\} \subseteq X$, consisting of at most $2d$ elements, can be identified as X because both sets span the same Rect(X). We obtain Ω_{2d} by collecting such X_{2d} for every $X \in \Omega$. Here, $|\Omega_{2d}| = |\Omega|$. Next, we show that Ω_{2d} is also Sperner family. Let us assume that $X_{2d} \subseteq X'_{2d}$. Then Rect$(X_{2d}) \subseteq$ Rect(X'_{2d}) and, thus, Rect$(X) \subseteq$ Rect(X'). This means $X \subseteq X'$ for $X, X' \in \Omega$. This is a contradiction. Thus, a Sperner family Ω_{2d} consists of subsets, each of which has at most $2d$ elements. By Corollary 1, we obtain the theorem. □

This estimation is useful when the value of d is small enough compared with the number of positive samples n.

4.2. *Evaluation by a criterion of PAC learning*

We evaluate the performance of the randomized algorithm, using a criterion of a PAC (Probably Approximately Correct) paradigm [for example, see reference (10)]. Let Ω be the true concept and Θ a concept estimated by an algorithm. In our case, Ω is the subclass family on S^+ and Θ is an output from a learning algorithm. Then, the algorithm is required to satisfy

the following criterion:

$$\Pr\{P(\Omega\Delta\Theta) \geq \varepsilon\} \leq \delta.$$

Here, $\Omega\Delta\Theta$ denotes the *symmetric difference* of Ω and Θ, that is, the set of elements belonging to only one of Ω or Θ, and $P(\Omega\Delta\Theta)$ denotes the probability of $\Omega\Delta\Theta$ measured by a probability $P(\cdot)$ defined on a universal set. That is, this criterion requires that with a high reliability of at least $(1-\delta)$, the algorithm outputs a good approximation with an accuracy of at least $(1-\varepsilon)$ for the true concept. A concept Ω is said to be *polynomial-sample learnable* if there exists a learning algorithm which requires the number of samples s to be polynomial with respect to $1/\varepsilon$, $1/\delta$ and n in order to attain the criteria, for all probability distributions P on the universal set. Samples are given randomly according to P by calling EXAMPLE oracle.

Our problem is a special case of PAC learning: for a given instance (S^+, S^-) of **SC-RECT**(n, m, d), the universal set is the power set of the power set of S^+, i.e. $2^{2^{S^+}}$ and the concept to be learned is the subclass family $\Omega(S^+, S^-)$. Moreover, we consider only P such that it is nonzero only on Ω. Samples included in Ω, i.e. subclasses, are given randomly according to P in response to calls of EXAMPLE. The degree of approximation is also measured by the same P.

Then, we have Theorem 3.

Theorem 3. There exists a learning algorithm that requires s samples (subclasses) in order to satisfy the PAC criterion, where:

$$s(\varepsilon, \delta, \Omega) = \frac{1}{\varepsilon}\left(|\Omega| \cdot \ln(2) + \ln\left(\frac{1}{\delta}\right)\right),$$

where ln denotes the natural logarithm.

Proof. The learning algorithm is as follows; it makes s calls of EXAMPLE and outputs the set of all obtained samples. This is enough because only samples from Ω, i.e. only subclasses, are generated. Let $\mathscr{F}_\Omega$ be the power set of Ω. The algorithm always outputs a subset Θ of Ω. Thus, $\Theta \subseteq \Omega$ and $\Theta, \Omega \in \mathscr{F}_\Omega$. We can prove this theorem in the same way as that described in the proof of Theorem 2.1 in reference (10). Let us assume that $P(\Theta\Delta\Omega) \geq \varepsilon$. For a particular Θ, the probability that any call of EXAMPLE will produce a subclass included in Θ is at most $(1-\varepsilon)$. Hence, the probability that s calls of EXAMPLE will produce only subclasses included in Θ is at most $(1-\varepsilon)^s$. It follows that the probability of s calls of EXAMPLE producing only subclasses included in any such choice of Θ is at most $(1-\varepsilon)^s$ summed over all the choices for Θ. There are at most $|\mathscr{F}_\Omega|$ choices for Θ. Hence, this probability is at most $|\mathscr{F}_\Omega|(1-\varepsilon)^s$. We will make s sufficiently large to bound this probability by δ. That is:

$$|\mathscr{F}_\Omega|(1-\varepsilon)^s \leq \delta.$$

Since:

$$|\mathscr{F}_\Omega| \leq 2^{|\Omega|},$$

we need

$$2^{|\Omega|}(1-\varepsilon)^s \leq \delta.$$

Taking natural logarithms on both sides of the inequality, we obtain:

$$|\Omega| \ln(2) + s\ln(1-\varepsilon) \leq \ln(\delta).$$

Using the approximation $\ln(1+\alpha) \leq \alpha$, it suffices if:

$$|\Omega| \ln(2) - s\varepsilon \leq \ln(\delta).$$

Simplifying, we have:

$$s \geq \frac{1}{\varepsilon}[|\Omega| \ln(2) - \ln(\delta)].$$

□

This theorem says that the algorithm needs s subclasses generated randomly according to a fixed P for obtaining a good approximation to a particular $\Omega(S^+, S^-)$. However, what we really want is the number of subclasses independent of Ω, that is $s(\varepsilon, \delta, n, d)$ instead of $s(\varepsilon, \delta, \Omega)$. From Theorem 2, we know that $|\Omega| \leq \binom{n}{2d} \leq n^{2d}$ for $2d < \lfloor n/2 \rfloor$. With Theorem 3, we have corollary 2.

Corollary 2. A subclass problem **SC-RECT**(n, m, d), $\lfloor n/2 \rfloor > 2d$, is polynomial-sample PAC learnable with s samples (subclasses), where:

$$s(\varepsilon, \delta, n, d) = \frac{1}{\varepsilon}\left(n^{2d}\ln(2) + \ln\left(\frac{1}{\delta}\right)\right).$$

Although this gives us an upper bound of the number of subclasses s needed for solving **SC-RECT**(n, m, d) approximately, it depends strongly on the upper bound n^{2d} of $|\Omega(S^+, S^-)|$ for instances $(S^+, S^-) \in$ **SC-RECT**(n, m, d). We believe that there exists a stricter upper bound of $|\Omega|$. Indeed, in $d = 2$, we have not so far been able to find any subclass family larger than $2n$, which is quite less than the estimation n^4.

4.3. *Connection between $P(X)$ and importance of X.*

The PAC criterion holds for any probability P if it is fixed and zero on the outside of Ω, that is, if only subclasses are given randomly according to P. However, the following problems remain: how to determine a proper P and how to generate subclasses randomly according to P. It is difficult to determine P such that higher values of P are assigned to more important subclasses and to generate subclasses according to such a P.

Our randomized algorithm realizes P on the basis of a uniform distribution over permutations. $P(X)$ is the probability that X is produced by the basic procedure for an equal occurrence of all $n!$ permutations of n positive samples. Thus, we will investigate the relation between such P and the degree of importance of subclasses.

Let us calculate P for a simple example shown in Fig. 3, where there exist four subclasses over seven positive samples.

In Fig. 3 let us calculate $P(X_1)$ for a subclass X_1. When an element $\mathbf{x}$ is taken from $S^+ = \{\mathbf{x}_1, \ldots, \mathbf{x}_7\}$

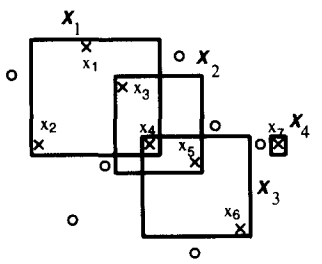


Fig. 3. An illustrative-example for the calculation of $P(X)$. There are four subclasses X_1–X_4.

firstly in the randomized algorithm, if $\mathbf{x}$ is either $\mathbf{x}_1$ or $\mathbf{x}_2$, then X_1 will be produced regardless of the following selection of elements, as they are included only in X_1. The probability that such an event occurs is $\frac{2}{7}$. If $\mathbf{x}$ is not any of $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_6$ and $\mathbf{x}_7$, the second element $\mathbf{x}'$ becomes a key to determine one from possible subclasses. If $(\mathbf{x}, \mathbf{x}')$ is $(\mathbf{x}_3, \mathbf{x}_1)$, $(\mathbf{x}_3, \mathbf{x}_2)$, $(\mathbf{x}_4, \mathbf{x}_1)$ or $(\mathbf{x}_4, \mathbf{x}_2)$ then X_1 will be produced independently of further elements. It should be noted that, at this stage, for the first element of $\mathbf{x}_3$, possible pairs are $(\mathbf{x}_3, \mathbf{x}_1)$, $(\mathbf{x}_3, \mathbf{x}_2)$, $(\mathbf{x}_3, \mathbf{x}_4)$ and $(\mathbf{x}_3, \mathbf{x}_5)$. Pairs $(\mathbf{x}_3, \mathbf{x}_6)$ and $(\mathbf{x}_3, \mathbf{x}_7)$ are impossible because they destroy exclusiveness. If such cases happen, the algorithm discards $\mathbf{x}_6$ or $\mathbf{x}_7$ and takes another element as the second element. Thus, in the case that the second element determines X_1, the probability is $\frac{1}{7}(\frac{2}{4} + \frac{2}{5})$. In the case that the third element determines the production of X_1, possible triplets are $(\mathbf{x}, \mathbf{x}', \mathbf{x}'') = (\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_1)$, $(\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_1)$, $(\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_2)$ and $(\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2)$. The probability is $\frac{1}{7}(\frac{1}{4} + \frac{1}{5})\frac{2}{3}$. Since the first three elements determine perfectly which subclass will be produced in this example, we do not need to examine further possibilities. In total, the probability that X_1 will be produced is 0.4571. Similarly, $P(X_2) = 0.1429$, $P(X_3) = 0.2571$ and $P(X_4) = 0.1429$.

Since the calculation is very complicated, we consider the following approximational probability:

$$P'(X) = \frac{1}{n}\left(\sum_{\mathbf{x} \in X} \frac{1}{D(x)}\right),$$

where $D(\mathbf{x})$ denotes the number of subclasses including $\mathbf{x}$. For example, $D(\mathbf{x}_1) = 1$, $D(\mathbf{x}_3) = 2$ and $D(\mathbf{x}_4) = 3$. When $P'(X)$ is used, $P'(X_1) = 0.4048$, $P'(X_2) = 0.1905$, $P'(X_3) = 0.2619$ and $P'(X_4) = 0.1429$. It is easy to confirm $\sum_X P'(X) = 1$.

Using $P'(X)$, let us discuss the connection between $P(X)$ and the degree of importance of X. For a larger value of $P'(X)$ there are two possibilities: (1) X has many samples or (2) X has decisive samples, or both. Here, we say that a sample $\mathbf{x}$ is *decisive* when $D(\mathbf{x})$ is small enough. The former possibility suggests that large subclasses are produced with a high probability in the algorithm. The latter possibility suggests that subclasses possessing decisive samples are also produced with a high probability. As a smaller subclass tends to have decisive samples, such a small subclass may be produced with a high probability. In our example, X_4 is such a case, as $P(X_4)$ is almost equal to $P(X_2)$, while the size of X_4 is less than the size of X_2. Furthermore, our algorithm is expected to find out subclasses so that they cover all positive samples, if every positive sample is exclusive itself. This is because every positive sample is chosen as the first element with the probability of $1/n$. Thus, the probability that a produced subclass includes a certain positive sample is not less than $1/n$. Thus, if we examine a large number of permutations, it is expected that the subclasses produced will cover all positive samples. We can strengthen this strategy by reassigning higher probabilities than before to positive samples not chosen so far. However, in this paper, we concentrate on our simple probability which is constant in every permutation.

5. EXPERIMENTAL RESULTS

We compared three methods.

(1) A subclass method[2] (S).

(2) A subclass method with pre- and postprocessings[2,5] (P).

(3) A randomized algorithm (R).

In the P method, we controlled the degree of reduction in the preprocessing by $n' \leftarrow \log_\alpha n$, where n' is the reduced number and α is a reduction parameter. In the following experiments, we do not use our upper bound with respect to s indicated in Corollary 2 as we consider it is too large to carry out and, it depends on each problem, i.e. on each $|\Omega|$. Through the following experiments, we adopted $\varepsilon = \delta = 0.01$. Every simulation was carried out on SUN SparcStation IPX.

5.1. *A two-class problem with normal distributions*

We considered two classes such that one mean vector is separated from the other by 4.0 in d-dimensional space. One class has a mean $(0, 0, \ldots, 0)$ and covariance matrix I and the other $(4/\sqrt{d}, 4/\sqrt{d}, \ldots, 4/\sqrt{d})$ and covariance matrix I. We generated training samples according to two normal distributions with these parameters. We used $\alpha = 1.1$. Furthermore, by $|\Omega| = nd$, we estimated the value of s, that is, $s = \frac{1}{\varepsilon}(nd \ln(2) + \ln(\frac{1}{\delta}))$. Varying the number of training samples n in each class, as $n(=m) = 20, 50, 100$, and the number of features d, as $d = 2, 4, 6, 8, 10$, we investigated the time needed for obtaining a subset of subclasses Θ and evaluated the accuracy of Θ according to the following three indices:

(1) $|\Theta|$.

(2) $\sum_{X \in \Theta} |X|/|S^+|$.

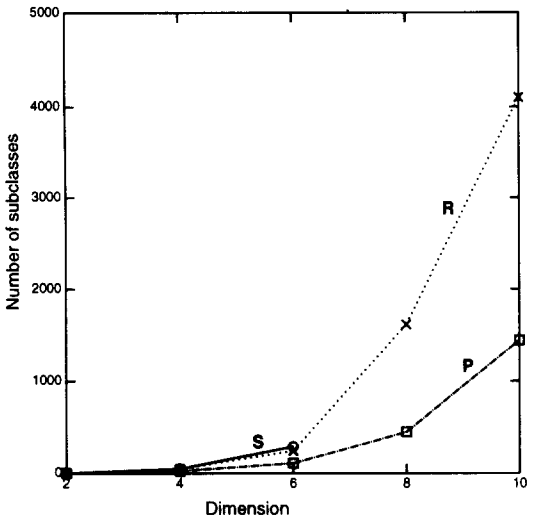
(3) $\max_{X \in \Theta} |X|/|S^+|$.

The first criterion shows how many subclasses were obtained. The second criteria tells us a total of sizes of

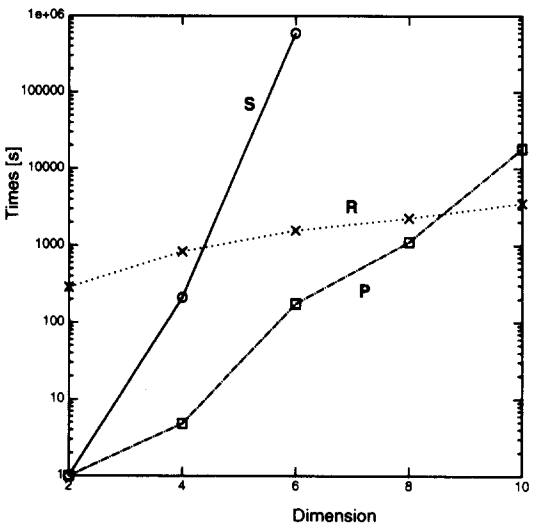
Table 1. Results of a two-class problem

Dimension (d)	Size (n)	Criterion: Num† S	Num† P	Num† R	Sum‡ S	Sum‡ P	Sum‡ R	Max§ S	Max§ P	Max§ R	Time(s)* S	Time(s)* P	Time(s)* R
2	20	1	1	1	1	1	1	1	1	1	0.0	0.0	0.0
	50	2	2	2	1.84	1.84	1.84	0.92	0.92	0.92	0.1	0.0	41.7
	100	2	2	2	1.87	1.87	1.87	0.95	0.95	0.95	0.2	0.0	285.2
4	20	3	3	3	2.7	2.7	2.7	0.95	0.95	0.95	0.0	0.0	11.6
	50	9	9	9	7.58	7.58	7.58	0.94	0.94	0.94	0.5	1.4	128.3
	100	49	25	49	29.12	16.83	29.12	0.88	0.88	0.88	210.1	4.8	834.9
6	20	1	1	1	1	1	1	1	1	1	0.0	0.0	0.4
	50	29	27	29	17.96	17.44	17.87	0.92	0.92	0.92	263.3	76.9	227.1
	100	295	110	248	147.84	67.43	134.95	0.82	0.82	0.82	590337.5	174.7	1561.4
8	20	1	1	1	1	1	1	1	1	1	0.0	0.0	0.4
	50	181	111	161	108.06	71.28	99.32	0.88	0.88	0.88	23078.1	1779.9	329.1
	100	—	452	1612	—	223.33	799.41	—	0.79	0.79	—	1106.6	2245.5
10	20	1	1	1	1	1	1	1	1	1	0.0	0.0	0.5
	50	580	372	422	744.54	203.92	239.14	0.86	0.86	0.86	301306.2	15659.9	493.2
	100	—	1443	4102	—	750.21	2210.62	—	0.79	0.79	—	18119.0	3511.6

* Time for construction of subclasses.
† $|\Theta|$.
‡ $\sum_{X\in\Theta}|X|/|S^+|$.
§ $\mathrm{Max}_{X\in\Theta}|X|/|S^+|$.
¶ S: a subclass method.
** P: a subclass method with pre- and postprocessings.
†† R: a randomized method. Any value related to the method is an average of five trials.
‡‡ The symbol "–" denotes that the calculation was terminated due to high cost.



(a)



(b)

Fig. 4. Results of a two-class problem ($n = 100$). Here, S is a subclass method, P is a subclass method with pre- and postprocessings and R is a randomized method.

subclasses as normalized by the size of the positive sample set. The last criterion tells us whether the most important subclass was obtained or not. Since the S method yields the optimal solution Ω, these values for Ω become maximum in all criteria.

The results are shown in Table 1 and Fig. 4. The results show that the R method is more effective than the P method, especially for large values of d, both in terms of time and accuracy.

5.2. *Alphabetical character recognition*

We treated 26 alphabetical handwritten uppercase letters from A to Z, taken from the ETL-3 database.[11] Each character is expressed by 342 ($= 19 \times 18$) bits (1: black pixel, 0: white pixel). The training sample set consisted of 2600 characters (100 for each character) and the test sample set consisted of another 2600 characters, that is, $n = 100$, $m = 2500$. We used 30 features ($d = 30$) based on characteristic loci as proposed by Glucksman.[12] As parameters, we used $\alpha = 1.2$ (100 samples into 25 samples) in the P method and $s = 1000$ in the R method. The number of obtained subclasses is shown in Table 2, along with the time needed for construction of subclasses and the discrimination rate. In discrimination, we used a small set of subclasses sufficient to cover all positive samples. An unknown sample is assigned to a class according to the class label attached to the largest hyper-rectangle, including the sample. If there is no such hyper-rectangle, the nearest hyper-rectangle to the sample is adopted. For details, see reference (5). The discrimination rates are almost the same. The R method found more subclasses in less time than the P method.

6. CONCLUSIONS

We proposed a randomized algorithm for approximating a region by a collection of hyper-rectangles, called subclasses. This algorithm yields a part of the set of all possible subclasses more efficiently than previous methods. The performance of approximation for the true subclass family is analysed by a criterion of PAC

Table 2. Results of character recognition

Method	Number of subclasses														
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P*	3	1	1	1	1	1	7	10	15	8	230	1	50	281	91
R†	12	1	1	1	1	1	11	9	32	10	364	1	195	436	218

Method	P	Q	R	S	T	U	V	W	X	Y	Z	Time(s)‡	Recognition rate (%)
P	1	1	8	70	5	1	1	1	1	28	1	13016	97.69
R	1	1	9	126	8	1	1	1	1	41	1	7250	97.65

* P: a subclass method with pre- and postprocessings.
† R: a randomized method.
‡ Time for construction of subclasses.

learning. Experimental results support the discussion. The most important problem yet to be clarified is to find a stricter upper bound for the number of subclasses. A randomized method which can generate subclasses in order of importance is also desirable.

Acknowledgement—The authors wish to thank Mr J. Toyama for his helpful discussions.

REFERENCES

1. M. Ichino, A nonparametric multiclass pattern classifier, *IEEE Trans. Syst. Man Cybernet.* **9**, 345–352 (1979).
2. M. Kudo and M. Shimbo, Optimal subclasses with dichotomous variables for feature selection and discrimination. *IEEE Trans. Syst. Man Cybernet.* **19**, 1194–1199 (1989).
3. M. Ichino, Optimum feature selection for decision functions, *IEICE* **J72-D-II**, 49–55 (1989) (in Japanese).
4. M. Kudo and M. Shimbo, Feature selection based on the structural indices of categories, *Pattern Recognition* **26**, 891–901 (1993).
5. M. Kudo and M. Shimbo, Analysis of the structure of classes and its applications—subclass approach, *Current Topics in Pattern Recognition Research.* 1, Council of Scientific Information. 69–81 (1994).
6. L. G. Valiant, A theory of the learnable, *Commun. ACM* **27**(11), 1134–1142 (1984).
7. J. C. Stoffel, A classifier design technique for discrete variable pattern recognition problems, *IEEE Trans. Comput.* **23**, 428–441 (1974).
8. E. Sperner, Ein satz über untermenger einer endlichen menge, *Math. Z.* **27**, 544–548 (1928).
9. D. Lubell, A short proof of sperner's lemma, *J. Comb. Theory* **1**, 299 (1966).
10. B. K. Natarajan, *Machine Learning.* Morgan Kaufmann Publishers, San Mateo, California (1991).
11. ETL3, ETL character database. Electrotechnical Laboratory/Japanese Technical Committee for Optical Character Recognition (1993).
12. H. A. Glucksman, Classification of characteristic loci, *Proc. IEEE Comput. Conf.* 138–141 (1967).

About the Author—MINEICHI KUDO received his D. Eng. degree in information engineering from the Hokkaido University in 1988. Since 1994, he has been Associate Professor of Information Engineering at Hokkaido University, Sapporo, Japan. His current research interests include pattern recognition, image processing and computational learning theory.

About the Author—SHINICHI YANAGI was born in Hokkaido, Japan, in 1970. He received his B.E. degree in information engineering from Hokkaido University, Sapporo, in 1994. His research interest is in the field of pattern recognition.

About the Author—MASARU SHIMBO received his D. Eng. degree in mathematical engineering from the University of Tokyo, Japan in 1976. Since 1979, he has been Professor of Information Engineering at Hokkaido University, Sapporo, Japan. From 1980 to 1981, he had been Visiting Professor of Cornell University, Ithaca, NY, and in 1981 Senior Visiting Fellow of the Science Research Council in the United Kingdom. His current research interests include speech recognition, visual perception and the other human information processing.