Multiclass Visual Classifier Based on Bipartite Graph Representation of Decision Tables*

Kazuya Haraguchi[1], Seok-Hee Hong[2], and Hiroshi Nagamochi[3]

[1] Faculty of Science and Engineering, Ishinomaki Senshu University, Japan
kazuyah@isenshu-u.ac.jp
[2] School of Information Technologies, University of Sydney, Australia
shhong@it.usyd.edu.au
[3] Department of Applied Mathematics and Physics,
Graduate School of Informatics, Kyoto University, Japan
nag@amp.i.kyoto-u.ac.jp

Abstract. In this paper, we consider *K-class classification problem*, a significant issue in machine learning or artificial intelligence. In this problem, we are given a training set of samples, where each sample is represented by a nominal-valued vector and is labeled as one of the predefined K classes. The problem asks to construct a *classifier* that predicts the classes of future samples with high accuracy. For $K = 2$, we have studied a new *visual classifier* named *2-class SE-graph based classifier* (2-SEC) in our previous works, which is constructed as follows: We first create several *decision tables* from the training set and extract a bipartite graph called an SE-graph that represents the relationship between the training set and the decision tables. We draw the SE-graph as a two-layered drawing by using an *edge crossing minimization* technique, and the resulting drawing acts as a visual classifier. We can extend 2-SEC to K-SEC for $K > 2$ naturally, but this extension does not consider the relationship between classes, and thus may perform badly on some data sets. In this paper, we propose SEC-TREE classifier for $K > 2$, which decomposes the given K-class problem into subproblems for fewer classes. Following our philosophy, we employ edge crossing minimization technique for this decomposition. Compared to previous decomposition strategies, SEC-TREE can extract *any* tree as the subproblem hierarchy. In computational studies, SEC-TREE outperforms C4.5 and is competitive with SVM especially when K is large.

1 Introduction

1.1 Background

We consider a mathematical learning problem called *classification*, a significant research issue from classical statistics to modern research fields on learning theory and data analysis [1]. For positive integers m, m' $(m \leq m')$, let

* This work is partially supported by Grant-in-Aid for Young Scientists (Start-up, 20800045) from Japan Society for the Promotion of Science (JSPS).

Table 1. A training set $S = S_1 \cup S_2$ with $S_1 = \{s^1, s^2, s^3, s^4\}$ and $S_2 = \{s^5, s^6, s^7\}$ over three attributes with $D_1 = \{\texttt{yes}, \texttt{no}\}$, $D_2 = \{\texttt{high}, \texttt{med}\}$, and $D_3 = \{\texttt{high}, \texttt{med}, \texttt{low}\}$

		Att. 1 (headache)	Att. 2 (temperature)	Att. 3 (blood pressure)
S_1 (malignant)	s^1	yes	high	high
	s^2	yes	med	med
	s^3	yes	high	high
	s^4	no	high	med
S_2 (benign)	s^5	no	med	high
	s^6	yes	high	med
	s^7	no	med	low

$[m] = \{1, 2, \ldots, m\}$ and $[m, m'] = \{m, m+1, \ldots, m'\}$. A *sample* s is represented by an n-dimensional nominal vector for n *attributes* and belongs to one of the predefined K *classes*. We denote the domain of attribute $i \in [n]$ by D_i, and the set of K classes by $C = \{c_1, \ldots, c_K\}$. In classification, we are given a *training set* $S = S_1 \cup \cdots \cup S_K$ of available samples, where S_k ($k \in [K]$) is the set of available samples belonging to the class c_k. Table 1 shows an example of training set for $K = 2$. A *classifier* is a function from the *sample space* $\mathcal{S} = D_1 \times \cdots \times D_n$ to the class set C. The aim of classification is to construct a classifier that predicts the class of a future sample $s' \in \mathcal{S}$ with high accuracy, where s' is possibly not in the training set, i.e., $s' \notin S$.

In our previous research, we have worked on developing a new *visual classifier* which can provide us with insights into data, beyond a mathematical function. Our main idea is to construct a classifier by good visualization of a graph extracted from the abstract data.

Visualization plays an important role as an effective analysis tool for huge and complex data sets in many application domains such as financial market, computer networks, biology and sociology [2]. Due to its popular application for visualization, graph drawing has been extensively studied over the last twenty years [3]. To draw graphs automatically and nicely, we need to give a mathematical definition of aesthetic criteria (e.g., the number of edge crossings) for 2D and 3D drawings. In our companion papers [4, 5], we proposed a new mathematical measurement of *occlusion* for $2.5D$ *drawing* [6] of pairwise trees, and observed that, when the samples and the attribute values are represented by pairwise trees, the minimum occlusion drawing supports visual analyses of classification and clustering. Independently, it was shown that algorithms for reducing edge crossings in graph drawings can be used in such data analyses as *rank aggregation* [7]. Based on these, we hypothesize that good visualization (e.g., visual objects with low visual complexity) itself can discover essential or hidden structure of data without relying on data analysis techniques, which can lead to a novel learning technique.

For 2-class classification (i.e., $K = 2$), we designed a prototype visual classifier and demonstrated its effectiveness by empirical studies in our preliminary

Table 2. Decision tables $T_1 = (A_1, \ell_1)$, $T_2 = (A_2, \ell_2)$ and $T_3 = (A_3, \ell_3)$ with attribute sets $A_1 = \{1, 2\}$, $A_2 = \{1, 3\}$ and $A_3 = \{3\}$

$T_1 = (A_1, \ell_1)$

$v \in D_1 \times D_2$	$\ell_1(v)$
`yes, high`	c_1
`yes, med`	c_1
`no, high`	c_1
`no, med`	c_2

$T_2 = (A_2, \ell_2)$

$v \in D_1 \times D_3$	$\ell_2(v)$
`yes, high`	c_1
`yes, med`	c_1
`yes, low`	c_1
`no, high`	c_2
`no, med`	c_1
`no, low`	c_2

$T_3 = (A_3, \ell_3)$

$v \in D_3$	$\ell_3(v)$
`high`	c_1
`med`	c_1
`low`	c_2

research [8, 9]. Recently, we found that our visual classifier is a visualization of *2-class generalized majority vote* (2-GMV) of multiple *decision tables* [10, 11].

Table 2 shows three decision tables for the training set in Table 1. Formally, a decision table $T = (A, \ell)$ is a classifier defined by a subset $A = \{i_1, \ldots, i_q\} \subseteq [n]$ of n attributes and a label function $\ell : D_{i_1} \times \cdots \times D_{i_q} \to C$. The label ℓ is often defined by *decision table majority* [12] in the literature, and in this paper, we do not assume any particular definition of ℓ. For a future sample $s \in \mathcal{S}$, a decision table estimates its class as the label $\ell(v)$ of the matched entry $v = s|_A$ (where $s|_A$ denotes the projection of s onto A). For example, (`no`, `high`, `high`) is classified as c_1 by T_1, c_2 by T_2 and c_1 by T_3. Let $\mathcal{T} = \{T_1, \ldots, T_N\}$ denote a set of N decision tables. One can use $\mathcal{T}$ as a single classifier by applying such ensemble technique as majority vote (MV). The MV classifies a future sample as the majority class among the N outputs given by the N decision tables in $\mathcal{T}$. For example, MV with $\mathcal{T} = \{T_1, T_2, T_3\}$ in Table 2 classifies (`no`, `high`, `high`) as c_1. GMV is a generalization of MV, and will be mentioned in Sect. 2.

For given S, $\mathcal{T}$ and its 2-GMV, our visual classifier in [10, 11] is built on a bipartite graph called *sample-entry graph* (SE-graph). In SE-graph, one node set corresponds to the samples in S, the other set corresponds to the entries of the decision tables in $\mathcal{T}$, and a sample s and an entry v are joined by an edge if and only if s matches v. See Fig. 1(a) for an example of SE-graph drawn as two-layered drawing. The layout of SE-graph is defined by permutations on the samples and the entries. We fix the permutation of the entry nodes along with the 2-GMV, and perform edge crossing minimization technique to compute the permutation on the sample nodes. We then divide the sample nodes into c_1 and c_2 sides by choosing a suitable threshold θ. (See θ dividing the sample nodes in Fig. 1(b).) A future sample s' will be judged as c_1 or c_2 by which side it falls on. We call this visual classifier *2-class SE-graph based classifier* (2-SEC).

1.2 New Contribution

In this paper, we extend 2-SEC to K-SEC for K-class problems ($K > 2$). The extension itself is not a hard task due to the following two reasons: One reason is that 2-GMV, the base classifier of 2-SEC, can be extended to K-GMV for

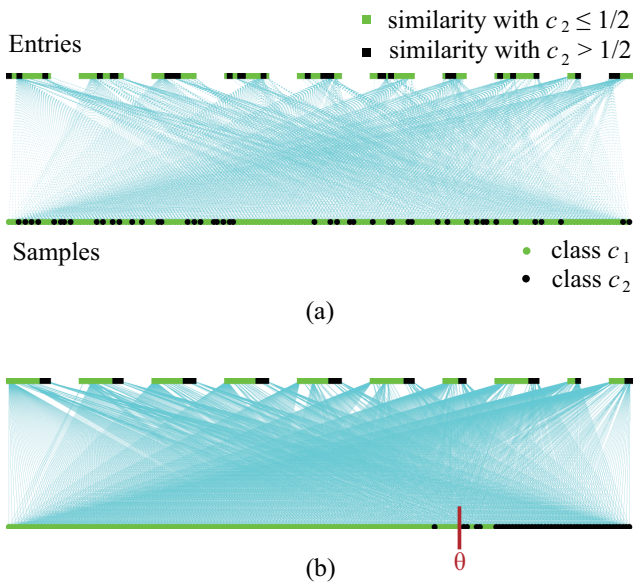

Fig. 1. Construction of 2-class SEC: (a) A natural SE-graph (b) The SE-graph obtained by edge crossing minimization

K-class problems naturally. We can visualize K-GMV as K-SEC, and its details are described in Sect. 2. The other reason is that we already know some general frameworks to extend any 2-class classifier to multiclass one. One may find the following three methods in the literature, *one-to-all*, *one-to-one*, and *error correcting output codes* [13]. These frameworks hardly take into account the structural relationships of classes, although it must be smarter to decompose the entire problem into subproblems for fewer classes in some application domains.

There are some studies that attempt to extract hierarchical structure of classes, which we call a *class tree*. Most of the previous works concentrate on extracting a binary tree as the class tree to decompose a K-class problem into 2-class subproblems (e.g., [14, 15]). In a class tree, there are K leaves, and each leaf corresponds to one of the K classes. (Figure 2 shows a class tree of Glass data set from UCI Repository [16].) Each inner node corresponds to a *meta-class*, representing the set of its descendant classes (i.e., leaves). Let $K' \leq K$ denote the number of children of an inner node. For this inner node, K'-class classifier is constructed, where each child constitutes one class. Starting from the root, a future sample is passed to one of the K' children, which is decided by the K'-class classifier of the current node. This procedure is repeated until the sample reaches a leaf. Finally, the sample is classified into the class of the reached leaf.

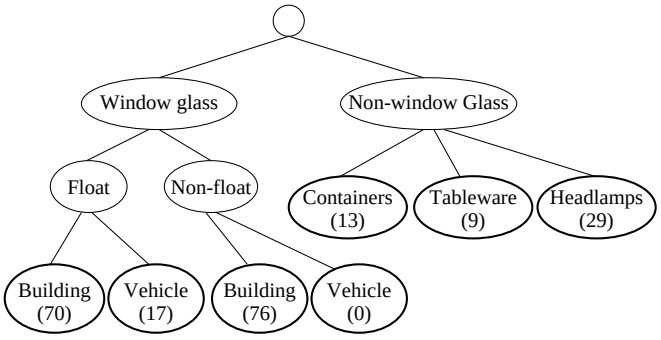

Fig. 2. Hierarchical structure of classes in GLASS data set: a leaf corresponds to a class and an integer represents the number of samples belonging to each class.

Note that one should extract a nice class tree and decompose the given K-class problem into easier subproblems. In the context of class tree, K-SEC can be regarded as a star (i.e., a tree consists only of the root and the K leaves), where no decomposition is made, in the sense that K classes are treated homogeneously in classifier construction. However, K-SEC does not seem to work well on such data sets that have structural relationships between classes. On the other hand, binary tree based approaches do not always work well because binary tree is not the universal structure of classes.

In this paper, we propose a new multiclass visual classifier, named SEC-TREE, that can extract *any* tree as a class tree. To extract a class tree, we employ edge crossing minimization on two-layered drawing of a bipartite graph, following our philosophy described in the previous subsection. We can control the structure of the resulting class tree by tuning the newly introduced parameter. In the extracted class tree, we use K'-SEC as the classifier of an inner node with K' children. When the number K of classes is large, in the sense of classification performance (i.e., error rate), SEC-TREE is superior to C4.5 [17], a standard decision tree classifier, and is competitive with well-tuned support vector machines (SVMs) for multiclass problems.

The paper is organized as follows. In Sect. 2, we introduce K-SEC and review 2-SEC as a special case of K-SEC. We describe how to construct SEC-TREE in Sect. 3. We present computational results in Sect. 4. Besides comparison of error rates, we show that our new method can capture a good class tree and discuss the effects of the controlling parameter.

Throughout this paper, we assume that a set $\mathcal{T}$ of decision tables is given. We do not focus on how to construct it in this paper. (In the experiments of Sect. 4, we utilize the decision tables generated by Weka [18].) This issue will be addressed in our future papers.

2 SE-graph Based Classifier (SEC)

Assume that a set $\mathcal{T} = \{T_1, \ldots, T_N\}$ of N decision tables is given. We denote $T_j = (A_j, \ell_j)$ $(j \in [N])$ and the set of all entries of T_j by $\mathcal{D}_j$, i.e., $\mathcal{D}_j$ is the Cartesian product of the domains of the attributes in A_j. For $K \geq 2$, K-GMV is defined by a tuple $(\lambda_1, \ldots, \lambda_N)$ of N *similarity functions* for the N decision tables. A similarity function λ_j is such a function $\lambda_j : \mathcal{D}_j \times C \to [0, 1]$ that represents "similarity" or "closeness" between each entry $v \in \mathcal{D}_j$ and class $c_k \in C$. For any entry $v \in \mathcal{D}_j$, we assume $\sum_{k=1}^{N} \lambda_j(v, c_k) = 1$. For a future sample $s \in \mathcal{S}$, K-GMV classifies s into the class c_k if *barycenter* $\beta(s, c_k)$ for c_k is the largest among all K classes $c_1, \ldots, c_K$, which is defined based on similarity functions as follows.

$$\beta(s, c_k) = \frac{1}{N} \sum_{j=1}^{N} \lambda_j(s|_{A_j}, c_k). \tag{1}$$

One can readily see that K-MV is a special case of K-GMV as follows; for each $(v, c_k) \in \mathcal{D}_j \times C$, set $\lambda_j(v, c_k) = 1$ if $\ell_j(v) = c_k$, and $\lambda_j(v, c_k) = 0$ otherwise.

Let us focus on $K = 2$. Since $\lambda_j(v, c_1) + \lambda_j(v, c_2) = 1$, it holds that $\beta(s, c_1) + \beta(s, c_2) = 1$. Whether s is classified into c_1 or c_2 is determined by whether $\beta(s, c_2) \leq 0.5$ or not. However, we do not need to stick to this threshold 0.5, and can replace it with any threshold $\theta \in [0, 1]$. For $K = 2$, let us define 2-GMV by a tuple $(\lambda_1, \ldots, \lambda_N, \theta)$.

Now we review 2-SEC, which serves as a visualization of 2-GMV [10, 11]. Let us denote the union of all $\mathcal{D}_j$'s by $\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_N$. A sample-entry graph (SE-graph) is a bipartite graph $G = (S, \mathcal{D}, E)$, which is defined as follows.

- S and $\mathcal{D}$ are node sets in the bipartite graph. Each sample $s \in S$ and each entry $v \in \mathcal{D}$ are nodes in a topological sense.
- For each $j \in [N]$, a sample $s \in S$ and an entry $v \in \mathcal{D}_j$ are joined by an edge (s, v) if and only if s matches the entry v. Thus the edge set is given by $E = E_1 \cup E_2 \cup \cdots \cup E_N$ such that $E_j = \{(s, v) \in S \times \mathcal{D}_j \mid s|_{A_j} = v\}$.

We consider the two-layered drawing of SE-graph in a 2D plane, i.e., one layer for S and the other for $\mathcal{D}$, as shown in Fig. 1. We order the entries from the same decision table consecutively, and take the permutation of the N decision tables arbitrarily. We define a *layout* of SE-graph by $(\sigma, \pi_1, \ldots, \pi_N)$, where $\sigma : S \to [|S|]$ is an *ordering* on the samples in S, and $\pi_j : \mathcal{D}_j \to [|\mathcal{D}_j|]$ $(j \in [N])$ is an ordering on the entries in $\mathcal{D}_j$.

Let $G_j = (S, \mathcal{D}_j, E_j)$ denote the subgraph of G induced by S and $\mathcal{D}_j$. We say that two edges $(s, v), (s', v') \in E_j$ *cross* if and only if $(\sigma(s) > \sigma(s')$ and $\pi_j(v) < \pi_j(v'))$ or $(\sigma(s) < \sigma(s')$ and $\pi_j(v) > \pi_j(v'))$. We define $x(G; \sigma, \pi_1, \ldots, \pi_N)$ as the sum of the edge crossings over all the N induced subgraphs;

$$x(G; \sigma, \pi_1, \ldots, \pi_N) = \sum_{j=1}^{N} \left| \{(s, s') \in S \times S \mid \sigma(s) < \sigma(s'),\ \pi_j(s|_{A_j}) > \pi_j(s'|_{A_j})\} \right|. \tag{2}$$

From the viewpoint of good visualization, we should determine the layout so that $x(G; \sigma, \pi_1, \ldots, \pi_N)$ is minimized. (We do not need to consider the crossings between edges from *different* $\mathcal{D}_j$'s since its number becomes a constant, but the reason is omitted due to space limitation.) We can compute the layout along with a given 2-GMV $(\lambda_1, \ldots, \lambda_N, \theta)$. Consider the *one-sided edge crossing minimization problem* (1CM) that asks to decide σ that minimizes $x(G; \sigma, \pi_1, \ldots, \pi_N)$, where $\pi_1, \ldots, \pi_N$ are fixed in the nondecreasing order of similarity function values, i.e., by denoting $\mathcal{D}_j = \{v_1, v_2, \ldots, v_B\}$ and $\lambda_j(v_1, c_2) \leq \lambda_j(v_2, c_2) \leq \cdots \leq \lambda_j(v_B, c_2)$, we assign $\pi_j(v_1) = 1$, $\pi_j(v_2) = 2$, $\ldots$, $\pi_j(v_B) = B$.

Since the 1CM problem is NP-hard [19], we utilize *barycenter heuristic* [20]. This heuristic permutes S in the nondecreasing order of barycenter $\beta(s, c_2)$ for class c_2. Computing its layout by barycenter heuristic, we assert that the SE-graph achieves a good visualization of the 2-GMV in the following sense:

(i) The number of edge crossings is (approximately) minimized. Indeed, barycenter heuristic has been recognized as an effective approximation algorithm in practice (e.g., [21]).

(ii) The resulting drawing can enable some meaningful analysis on 2-GMV. For example, the computed string of samples is split into two substrings according to whether barycenter is larger than θ or not. The samples in the former (resp., latter) substring are estimated as c_2 (resp., c_1) by the 2-GMV.

What we call 2-SEC is the visual classifier consisting of 2-GMV and its SE-graph.

For $K > 2$, a K-GMV $(\lambda_1, \ldots, \lambda_N)$ is visualized as K-SEC in a similar way. We construct K copies $G^1, \ldots, G^K$ of the SE-graph $G = (S, \mathcal{D}, E)$, one for each class $c_k \in C$. We draw each G^k by two-layered drawing in a 2D plane. The layout of G^k is computed as follows: The entries v's from decision table T_j are ordered by $\lambda_j(v, c_k)$, and the samples s's are ordered by barycenter heuristic based on $\beta(s, c_k)$.

3 SEC-TREE for Multiclass Classification

Let U denote any set of elements and 2^U denote its power set. A family $\mathcal{U} \subseteq 2^U$ is called *laminar* if, for any two sets $X, Y \in \mathcal{U}$, at least one of the three sets $X \setminus Y$, $Y \setminus X$, $X \cap Y$ is empty. It is obvious that a laminar family $\mathcal{U}$ can be visualized by a tree, where a node corresponds to $X \in \mathcal{U}$ and an edge represents the inclusion relationship. In the sequel, we may refer to a laminar family $\mathcal{U}$ as a tree and an $X \in \mathcal{U}$ as a node if no confusion arises.

To construct a class tree for the given K-class problem, we compute a laminar family $\mathcal{C} \subseteq 2^C$ of subsets of the class set C, and utilize $\mathcal{C}$ as the class tree. We include a subset $X \subseteq C$ in $\mathcal{C}$ if our criteria say that X should be treated as a meta-class in a 2-class (or at least fewer-class) subproblem. For this, we test if the samples of the classes in X and the samples of the classes in $\bar{X} = C \setminus X$ can be separated "effectively" by 2-SEC.

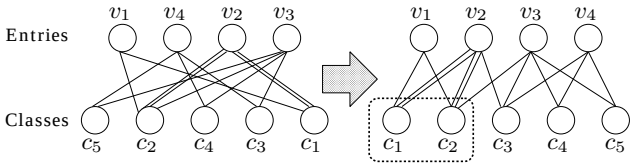

Fig. 3. Two-sided edge crossing minimization (2CM) on CE-graphs

To test the separability, we introduce *class-entry graph* (CE-graph), $\hat{G} = (C, \mathcal{D}, E)$, which is obtained from SE-graph $G = (S, \mathcal{D}, E)$ by contracting each $S_k \subseteq S$ ($k \in [K]$) into a node $c_k \in C$. We define a layout of CE-graph by $(\hat{\sigma}, \pi_1, \ldots, \pi_N)$, where $\hat{\sigma} : C \to [|C|]$ is an ordering on the class set and $\pi_j : \mathcal{D}_j \to [|\mathcal{D}_j|]$ ($j \in [N]$) is an ordering on the entry set $\mathcal{D}_j$. The number of edge crossings of CE-graph in the layout $(\hat{\sigma}, \pi_1, \ldots, \pi_N)$ is defined analogously with the case of SE-graph (see (2)).

Recall that 2-SEC is obtained as a result of performing 1CM on SE-graph. We consider *two-sided edge crossing minimization* (2CM) on CE-graph that asks to compute the layout $(\hat{\sigma}, \pi_1, \ldots, \pi_N)$ to minimize the edge crossings. See Fig. 3. In this simple example, we assume the number K of classes to be 5 and focus on one decision table with 4 entries. However, the discussion can be generalized easily. For convenience, let $\hat{\sigma}(c_1) = 1, \ldots \hat{\sigma}(c_K) = K$.

In the figure, we can expect that the samples of the classes in $X = \{c_1, c_2\}$ and the samples of the classes in $\bar{X} = \{c_3, c_4, c_5\}$ can be separated by an appropriate construction of 2-SEC. This expectation comes from the observation that there are few crossings between edges from X and $\bar{X}$, which may suggest the separability between X and $\bar{X}$, and that there are more edge crossings in inside of X, which may suggest that the samples of the classes in X match the similar entries (and thus take close values for barycenter).

Now let us formalize our criteria to decide whether $X \subseteq C$ should be included as a node in the class tree. For $k \in [K]$ and $t \in [K - k] \cup \{0\}$, let $X_{k,t} = \{c_k, c_{k+1}, \ldots, c_{k+t}\}$ denote a consecutive subset of C. We define $\chi(k, t)$ to be the number of crossings between edges from $X_{k,t}$ and $\bar{X}_{k,t}$, and define $\eta(k, t)$ as follows;

$$\eta(k,t) = \begin{cases} 0 & \text{if } t = 0, \\ \max_{[k',t'] \subsetneq [k,t]} \frac{\chi(k,t)}{\chi(k',t')} & \text{otherwise.} \end{cases} \tag{3}$$

One can see that, if $\eta(k, t)$ is small, then the crossings between edges from $X_{k,t}$ and $\bar{X}_{k,t}$ are relatively fewer than the edge crossings inside $X_{k,t}$.

Our algorithm to construct a class tree is described in Algorithm 1. The smallness of $\eta_{k,t}$ is decided by a positive parameter $\delta > 0$. In line 1, since the 2CM problem is NP-hard [22], we employ iterative application of barycenter heuristic in the experiments of Sect. 4, i.e., repeat fixing one side and permuting the other side by barycenter heuristic until no change is made on both sides. In

line 2, we can compute all $\eta(k,t)$'s efficiently by dynamic programming. However, the details are omitted due to space limitation.

Algorithm 1 ConstructClassTree

1: Compute the layout $(\hat{\sigma}, \pi_1, \ldots, \pi_N)$ of CE-graph by 2CM.
2: For each $k \in [K]$ and $t \in [K-k] \cup \{0\}$, compute $\eta(k,t)$ by (3).
3: $\mathcal{X} \leftarrow \{X_{k,t} \subseteq C \mid \eta(k,t) < \delta, k \in [K], t \in [K-k] \cup \{0\}\}$.
We denote $\mathcal{X} = \{X_{k_1,t_1}, \ldots, X_{k_b,t_b}\}$, where $\eta(k_a,t_a) \leq \eta(k_{a+1},t_{a+1})$ $(\forall a \in [b-1])$.
4: $\mathcal{C} \leftarrow \emptyset$.
5: **for** $a \leftarrow 1, 2, \ldots, b$ **do**
6: **if** $\mathcal{C} \cup \{X_{k_a,t_a}\}$ is laminar **then**
7: $\mathcal{C} \leftarrow \mathcal{C} \cup \{X_{k_a,t_a}\}$.
8: **end if**
9: **end for**
10: Output $\mathcal{C}$.

For leaves of the class tree, the output $\mathcal{C}$ always includes singletons $X_{1,0} = \{c_1\}, \ldots, X_{K,0} = \{c_K\}$ for any $\delta > 0$ since $\eta(k,0) = 0 < \delta$ by (3). It is possible that $\mathcal{C}$ contains more than one maximal subset, i.e., more than one class tree. In such a case, we have to choose the class tree used for classifying a future sample, but we omit the details due to space limitation. (In our preliminary experiments, we hardly observed such a case.)

The parameter δ eventually controls the structure of the output class tree. Intuitively, if $\delta \to 0$ (resp., $+\infty$), then less (resp., more) subsets are likely to be included in $\mathcal{C}$, and thus the class tree is close to a star (resp., a binary tree). Hence it is expected that a larger δ decomposes the given K-class problem into more subproblems for fewer classes. We will observe the effects of δ in the experiments described in Sect. 4.

For the computed class tree $\mathcal{C}$, let us mention how K'-SEC is constructed for an inner node $X \in \mathcal{C}$ $(|X| \geq 2)$ with K' children. Let $Y_1, \ldots, Y_{K'} \in \mathcal{C}$ denote the children of inner node X. For each decision table $T_j \in \mathcal{T}$, let $m_j(v, Y_k)$ denote the number of the samples matching entry $v \in \mathcal{D}_j$ whose class is in Y_k, and let $M_j(v) = \sum_{k=1}^{K'} m_j(v, Y_k)$. Then we employ the similarity function $\lambda_j(v, Y_k)$ with the following definition;

$$\lambda_j(v, Y_k) = \begin{cases} 0 & \text{if } M_j(v) = 0, \\ \frac{m_j(v,Y_k)}{M_j(v)} & \text{otherwise.} \end{cases}$$

When $K' = 2$, recall that we are free to choose the threshold $\theta \in [0,1]$. Aiming at improving the prediction performance, we set the threshold θ to the value that minimizes the error rate on the training set. Further, we employ *two-stage* 2-*SEC* as follows; we construct the first 2-SEC $(\lambda_1, \ldots, \lambda_N, \theta)$ from the training set S, and the second 2-SEC $(\lambda'_1, \ldots, \lambda'_N, \theta')$ from the subset $S' \subseteq S$, where S' is the subset of samples falling "near" the threshold θ of the first 2-SEC. A future sample is classified by the second 2-SEC if it falls near θ.

Table 3. Data sets from UCI Repository

Data	Classes (K)	Samples	Test	Attributes		
				(Num)	(Nom)	→ (Binary)
Lenses	3	24	(10-CV)	0	4	-
Iris	3	150	(10-CV)	4	0	6.7
Wine	3	178	(10-CV)	13	0	5.8
Balance	3	625	(10-CV)	4	0	16.0
Cmc	3	1473	(10-CV)	2	7	49.1
Car	4	1728	(10-CV)	0	6	-
Nursery	5	12960	(10-CV)	0	8	-
Bridges	6	105	(10-CV)	1	10	15.6
Dermatology	6	366	(10-CV)	1	33	12.1
Anneal	6	798	100	6	32	27.0
Sat	6	4435	2000	36	0	50
Zoo	7	101	(10-CV)	0	17	-
Glass	7	214	(10-CV)	9	0	14.4
Yeast	10	1484	(10-CV)	8	0	38.5
Soybean	19	307	376	0	35	-
Audiology	24	200	26	0	69	-

Finally, SEC-TREE is the visual classifier consisting of the class tree $\mathcal{C}$ and the K'-SEC's for the inner nodes. Let us emphasize that our class tree is constructed based on edge crossing minimization on CE-graph.

4 Computational Experiments

4.1 Experimental Settings

In the experiments, we use 16 data sets from UCI Machine Learning Repository [16]. Table 3 shows the summary of the data sets. We construct a classifier from the training set and evaluate it by the error rate on the *test set*. Anneal, Sat, Soybean and Audiology have their own test sets, and we use them for the evaluation. (For these data sets, the column "Test" shows the number of samples in the test set.) For the other data sets, we perform *10-fold cross validation* [23] to generate training and test sets.

The columns "(Num)" and "(Nom)" show the numbers of numerical and nominal attributes, respectively. SEC-TREE is formulated on nominal data sets, and it cannot handle a data set with numerical attributes; we need to transform the data set into nominal one by an appropriate algorithm. For this, we employ the algorithm proposed in our previous work [24], which extracts a set of binary attributes from a data set with numerical attributes. An extracted binary attribute may take 1 (resp., 0) if some numerical attribute value is (resp., is not) larger than a computed threshold. We extract a set of binary attributes from the training set, and then construct SEC-TREE from the binarized training set. The test set is also binarized by the same set of binary attributes, and is used for

classifier tests. The rightmost column "(Binary)" shows the number of extracted binary attributes, where the fractional number represents its average in 10-fold cross validation. Some data sets contain only nominal attributes (which are indicated by hyphen). For such data sets, we can construct SEC-TREE without binarization.

Let us describe how to construct a set $\mathcal{T} = \{T_1, \ldots, T_N\}$ of N decision tables. We use $N = 10$, 20 and 30. We generate the attribute set A_j of each decision table $T_j = (A_j, \ell_j)$ by DecisionTable package of Weka [18]. This package generates a "good" attribute set by *local search*. By choosing the initial solution at random, we can generate different attribute sets. See [12, 18] for the details.

4.2 Validity of Obtained Class Tree

In this subsection, we illustrate that our new method can find a class tree that is close to the one inherent in Glass data set. In this data set, a sample corresponds to a type of glass, the attribute values represent information on chemical ingredients, and the class represents the object where the glass is used. The manual of the data set says that the 7 classes form a hierarchy, as shown in Fig. 2.

For simplicity, we concentrate on constructing binary class trees; the parameter δ is set to $+\infty$ here. We investigate if the constructed class tree has either of the two meta-classes, "Window glass" and "Non-window glass." The class "Vehicle" under "Non-float" is ignored, since no sample belongs to the class.

For comparison, we introduce two types of binary class trees constructed in different ways. One is called *equally sized class tree*, which is constructed by partitioning the ordered class set (which is obtained by 2CM on CE-graph) recursively so that the difference in the number of the samples between the two children becomes so small as possible. The other is *random class tree*, which is constructed by partitioning the ordered class set at random recursively.

When $N = 10$ (resp., 20 and 30), our method finds either cluster by 66% (resp., 74% and 69%), while the equally sized class tree finds either one by 44% (resp., 40% and 45%) and the random class tree does by 25% (resp., 26% and 26%), which tells that our method succeeds in capturing nontrivial approximation of the true hierarchy.

For error rates, we cannot observe significant difference among the three class trees in our preliminary experiments. However, they are different in the depths of the nodes where misclassification occurs. We assert that misclassification in a shallow node costs more than one in a deep node because, in the former case, we may have to improve not only the K'-SEC's of the misclassifying inner nodes but also the class tree structure. Figure 4 shows the distribution of the misclassified samples for the depth difference between the misclassifying inner node and the leaf of the true class. In the proposed class tree, misclassified samples are more distributed on small differences than the others. Hence it is expected that SEC-TREE of the proposed class tree can be improved further by detailed tuning only on deeper nodes, which is left for future work. We remark that this phenomenon

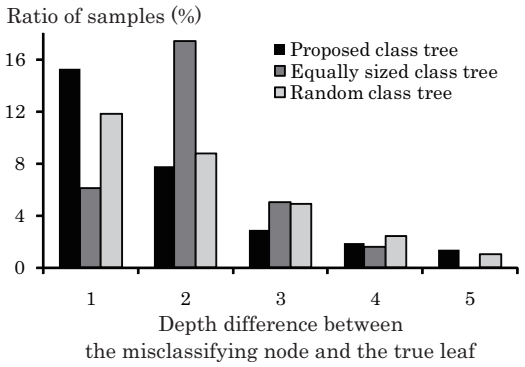

Fig. 4. Distribution of misclassified samples in test sets

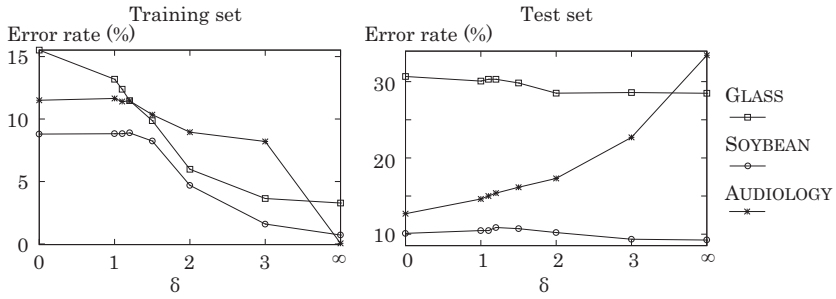

Fig. 5. Change of error rate along with δ (left: training set, right: test set)

is also observed in the cases of $N = 20$ and 30, and in most of the other data sets.

4.3 Parameter Effects

The parameter δ controls the degree to which the given K-class problem is decomposed. In Fig. 5, we observe the effects of δ in terms of error rate. The left figure shows the change of error rate on the training set of GLASS, SOYBEAN, and AUDIOLOGY. We use $N = 20$ here. As shown, SEC-TREE with a larger δ (where the problem is decomposed into more subproblems for fewer classes) fits to the training set better than a smaller δ, which may meet one's intuition. On the other hand, the right figure shows the change of error rate on the test set. When δ becomes large, the error rate for AUDIOLOGY increases rapidly, which shows *overfitting*, while the error rates for the other two data sets are improved by 2 to 3%. How the problem should be decomposed must depend on data sets, and we have to decide the value of δ carefully.

4.4 Comparison with Other Classifiers

We compare SEC-TREE with other classifiers, C4.5 [17], LibSVM [25] and MCSVM [26], in terms of error rate on test sets. All the classifiers have some tunable parameters. We try some combinations of parameter values for each classifier, and evaluate it by the smallest error rate. For SEC-TREE, we set δ to ε, 1.0, 1.1, 1.2, 1.5, 2.0, 3.0 and $+\infty$, where ε is a sufficiently small positive number. For C4.5, we test 8 combinations of parameter values: we set `confidence rate` to 1%, 25% (default), 50% or 99%, `binary split` option to `true` or `false` (default), and the other parameters to the default values. For LibSVM, we test 16 combinations: we use 2-class C-SVM and RBF kernel, and set $C = 0.5$, 1.0 (default), 2.0 or 4.0, $\gamma = 0.0$ (default), 0.5, 1.0 or 2.0, and the other parameters to the default values. Note that LibSVM employs one-to-one framework to extend 2-class C-SVM to multiclass one. Since MCSVM is an extension of 2-class C-SVM, LibSVM and MCSVM have similar parameters in common. For MCSVM, we test the same 16 combinations as LibSVM.

We show the results in Table 4. Boldface for each data set shows the smallest (i.e., best) error rate among all classifiers. A sign $*$ on SEC-TREE indicates that the error rate is smaller than C4.5. The effectiveness of SEC-TREE is outstanding when K is large; for $K \geq 7$, SEC-TREE outperforms C4.5 for all data sets and becomes more competitive with SVMs. In particular, SEC-TREE is much better than the other classifiers for AUDIOLOGY, which has the largest K among the used data sets. For larger K, we may have to decompose K-class problem more carefully. The experimental results indicate that SEC-TREE succeeds in extracting class trees which are effective in decreasing error rates.

Let us describe computation time briefly. We evaluate the actual time needed to construct a classifier and to classify a test set. All experiments are conducted by our PC with 2.83GHz CPU. C4.5 takes at most 2 seconds for all data sets. For SVMs and SEC-TREE, we observe NURSERY data set as an extreme case, since it has the largest number of samples. The computation time of SVMs heavily depends on parameter values; both LibSVM and MCSVM take from 10 to 350 seconds. We observe that the error rates of SVMs are also sensitive to parameter values. It must be exhaustive to search appropriate parameter values. On the other hand, SEC-TREE takes about 21 (resp., 35 and 73) seconds when $N = 10$ (resp., 20 and 30) regardless of δ. We note that more than 95% of the computation time is devoted to generation of $\mathcal{T}$, for which we use Weka as a subroutine. We expect that the computation time can be improved by developing a faster algorithm to generate $\mathcal{T}$.

5 Concluding Remarks

In this paper, we proposed a new multiclass visual classifier SEC-TREE as an extension of our previous 2-class classifier 2-SEC. SEC-TREE can extract any tree as the class tree by tuning the parameter δ. We presented computational results to show the effectiveness of the proposed method.

Table 4. Error rates (%) of SEC-TREE, C4.5, LibSVM and MCSVM

Data	Classes (K)	SEC-TREE			C4.5	LibSVM	MCSVM
		$N = 10$	20	30			
Lenses	3	20.33	20.00	20.17	16.66	21.66	**0.00**
Iris	3	5.80	5.80	5.80	4.66	**3.99**	4.67
Wine	3	10.47	10.69	10.74	9.08	**9.05**	**9.05**
Balance	3	∗13.55	∗13.23	∗13.13	20.16	**8.63**	9.17
Cmc	3	∗45.25	∗45.10	∗44.90	45.41	**44.05**	45.43
Car	4	3.30	3.35	3.46	2.83	**0.34**	1.24
Nursery	5	0.97	0.73	0.73	0.62	0.04	**0.03**
Bridges	6	42.71	41.05	40.42	39.09	**38.18**	39.27
Dermatology	6	∗14.33	∗14.27	∗14.32	14.99	12.53	**12.50**
Anneal	6	7.90	7.80	7.90	6.00	6.00	**4.00**
Sat	6	∗15.71	∗15.03	∗15.01	16.70	**12.15**	12.40
Zoo	7	∗**0.00**	∗**0.00**	∗**0.00**	1.00	1.00	**0.00**
Glass	7	∗29.47	∗28.28	∗27.37	32.72	**25.17**	25.36
Yeast	10	∗41.48	∗40.98	∗40.64	41.91	**40.22**	42.50
Soybean	19	∗10.77	∗9.23	∗9.76	12.76	**7.44**	9.19
Audiology	24	∗15.00	∗12.69	∗**11.92**	15.38	34.61	23.08

Our main future work is described as follows: **(i)** We have assumed that a set $\mathcal{T}$ of decision tables is given and generated it by Weka in the experiments. We need to develop a faster algorithm to generate a better $\mathcal{T}$. **(ii)** In the experiments, we used a binarization algorithm to deal with a data set with numerical attributes since our formulation is limited to nominal data sets. We should consider an extended formulation that can treat numerical attributes directly. **(iii)** We also have to find application areas where our visual classifier is effective for data analysis and knowledge discovery.

References

1. Friedman, J.H.: Recent advances in predictive (machine) learning. Journal of Classification **23** (2006) 175–197
2. Ware, C.: Information Visualization: Perception for Design. 2nd edn. Morgan Kaufman (2004)
3. Battista, G.D., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice Hall (1999)
4. Haraguchi, K., Hong, S., Nagamochi, H.: Visual analysis of hierarchical data using 2.5D drawing with minimum occlusion. poster session at IEEE PacificVis 2008.
5. Haraguchi, K., Hong, S., Nagamochi, H.: Visual analysis of hierarchical data using 2.5D drawing with minimum occlusion. Technical Report 2009-010, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Japan (2009)
6. Ware, C.: Designing with a 2 1/2D attitude. Information Design Journal **10**(3) (2001) 171–182

7. Dwork, C., Kumar, R., Naor, M., Sivakumar, D.: Rank aggregation methods for the web. In: WWW10, ACM (2001) 613–622
8. Haraguchi, K., Hong, S., Nagamochi, H.: Classification by ordering data samples. RIMS Kokyuroku **1644** (2009) 20–34 ISSN 1880-2818.
9. Haraguchi, K., Hong, S., Nagamochi, H.: Classification via visualization of sample-feature bipartite graphs. Technical Report 2009-011, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Japan (2009)
10. Haraguchi, K., Hong, S., Nagamochi, H.: Visualization can improve multiple decision table classifiers. In: Proc. MDAI. (2009) to appear.
11. Haraguchi, K., Hong, S., Nagamochi, H.: Bipartite graph representation of multiple decision table classifiers. In: Proc. SAGA. Volume 5792 of LNCS., Springer (2009) 46–60
12. Kohavi, R.: The power of decision tables. In: ECML. Volume 912 of LNAI., Springer (1995) 174–189
13. Dietterich, T.G., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research **2** (1995) 263–286
14. Kumar, S., Ghosh, J., Crawford, M.M.: Hierarchical fusion of multiple classifiers for hyperspectral data analysis. Pattern Analysis and Applications **5**(2) (2002) 210–220
15. Cheng, L., Zhang, J., Yang, J., Ma, J.: An improved hierarchical multi-class support vector machine with binary tree architecture. Proc. International Conference on Internet Computing in Science and Engineering (2008) 106–109
16. Asuncion, A., Newman, D.: UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences (2007) `http://www.ics.uci.edu/~mlearn/MLRepository.html`.
17. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
18. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. 2nd edn. Morgan Kaufmann, San Francisco (2005) `http://www.cs.waikato.ac.nz/ml/weka/`.
19. Eades, P., Wormald, N.C.: Edge crossings in drawings of bipartite graphs. Algorithmica **11** (1994) 379–403
20. Sugiyama, K., Tagawa, S., Toda, M.: Methods for visual understanding of hierarchical system structures. IEEE Transactions on Systems, Man, and Cybernetics **SMC-11**(2) (1981) 109–125
21. Jünger, M., Mutzel, P.: 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. Journal of Graph Algorithms and Applications **1**(1) (1997) 1–25
22. Garey, M.R., Johnson, D.S.: Crossing number is NP-complete. SIAM Journal on Algebraic and Discrete Methods **4** (1983) 312–316
23. Weiss, S.M., Kulikowski, C.A.: Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems. Morgan Kaufmann (1991)
24. Haraguchi, K., Nagamochi, H.: Extension of ICF classifiers to real world data sets. In: IEA/AIE. Volume 4570 of LNAI., Springer (2007) 776–785
25. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. (2001) Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
26. Crammer, K., Singer, Y.: On the algorithmic implementation of multiclass kernel-based vector machines. Journal of Machine Learning Research **2** (2001) 265–292