Mach Learn (2011) 85:249–272
DOI 10.1007/s10994-011-5240-0

# Ternary Bradley-Terry model-based decoding for multi-class classification and its extensions

**Takashi Takenouchi · Shin Ishii**

Received: 17 January 2010 / Revised: 15 October 2010 / Accepted: 2 February 2011 /
Published online: 27 February 2011

**Abstract** A multi-class classifier based on the Bradley-Terry model predicts the multi-class label of an input by combining the outputs from multiple binary classifiers, where the combination should be *a priori* designed as a code word matrix. The code word matrix was originally designed to consist of $+1$ and $-1$ codes, and was later extended into deal with ternary code $\{+1, 0, -1\}$, that is, allowing 0 codes. This extension has seemed to work effectively but, in fact, contains a problem: a binary classifier forcibly categorizes examples with 0 codes into either $+1$ or $-1$, but this forcible decision makes the prediction of the multi-class label obscure. In this article, we propose a Boosting algorithm that deals with three categories by allowing a 'don't care' category corresponding to 0 codes, and present a modified decoding method called a 'ternary' Bradley-Terry model. In addition, we propose a couple of fast decoding schemes that reduce the heavy computation by the existing Bradley-Terry model-based decoding.

**Keywords** Multi-class classification · Bradley-Terry model · Ensemble learning · Decoding

## 1 Introduction

Development of classification methods is one of the major research topics in the fields of machine learning and pattern recognition (Hastie et al. 2001). While methods for binary classification, such as classical linear discriminant analysis (LDA), support vector machine (SVM) (Vapnik 1995) and AdaBoost (Freund and Schapire 1997), are well established,

Editor: Phil Long.

T. Takenouchi (✉)
Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan
e-mail: ttakashi@is.naist.jp

S. Ishii
Graduate School of Informatics, Kyoto University, Kyoto, Japan

multi-class classification remains challenging, and methods to achieve it are still being developed. There are two major types of approaches to multi-class classification problems. One tries to directly construct discriminant functions for multiple classes, for example, by estimating conditional probability density for each class. Recently, several multi-class extensions of SVM (Weston and Watkins 1999; Crammer and Singer 2001) and AdaBoost (AdaBoost-M2 and so on) (Freund and Schapire 1997) have also been proposed. The other approach decomposes an original multi-class problem into multiple binary classification problems and rather indirectly makes a multi-class discriminant function so as to integrate results of constituent binary classifiers.

In this study, we take the latter approach. Dietterich and Bakiri presented a general framework called error-correcting output coding (ECOC), in which an original multi-class problem was decomposed into an arbitrary number of binary classification problems (Dietterich and Bakiri 1995), and the simplest decoding method, Hamming decoding, was employed to obtain a multi-class decision. Although the original ECOC did not allow the code matrix to contain zero components, Allwein et al. extended the ECOC framework to allow the code matrix to have zero components and presented a more general loss-based decoding (Allwein et al. 2001). A decoding method based on probabilistic modeling of the noise process constituted by the trained binary classifiers was also proposed by the authors (Takenouchi and Ishii 2009).

On the other hand, Hastie and Tibshirani presented a different approach to the ECOC decoding problem (Hastie and Tibshirani 1998), which integrates the results of binary classifiers designed as the 1–1 code word matrix into a single estimate of class membership probabilities, based on a probabilistic model of 1–1 matches of two classes, called the Bradley-Terry model (BT) (Bradley and Terry 1952). Zadrozny (2001) extended this probabilistic decoding method so as to be applicable to arbitrary code matrices including zero components. We previously proposed an analog ECOC coding scheme whose code values are optimized based on the multi-class classification performance (Yukinawa et al. 2008).

Although these BT model-based approaches have seemed to work well, application of the BT model to ECOC in fact contains a big problem; a feature vector whose real code is 0 is forcibly categorized into either $+1$ or $-1$, but this meaningless decision is harmful because it may make decisions by other binary classifiers obscure. There have been several studies to deal with ternary categories, $+1$, 0 and $-1$. A simple two-stage approach was taken in Moreira and Mayoraz (1998): the first-stage discriminates whether an input feature vector has code 0 or not, and the second-stage discriminates whether an input has code $+1$ or $-1$. A tri-class SVM that applies constraints to feature vectors with code 0 was also proposed (Angulo and Català 2000; Angulo et al. 2006) and was employed for the multi-class classification. In Cutzu (2003), an influence of feature vectors with code 0 was implicitly considered in the decoding process. On the other hand, the BT model-based methods require an optimization process to obtain a probability membership estimate for each feature vector, (*i.e.* to perform decoding), whose implementation due to iterative calculations requires much computation time. In addition, when dealing with ternary code $+1, 0, -1$,[1] decoding of multi-class labels has often been implemented by a (weighted) voting scheme, which is simple but cannot consider the reliability of each binary classifier. A main objective of our current study is to propose a computationally feasible method for multi-class classification, based on an appropriate treatment of ternary code.

In this article, we propose a Boosting algorithm to deal with ternary code and a fast decoding method under the formulation of a 'ternary' BT model, in order to solve the above

[1] In this article, we call a triplet $\{+1, 0, -1\}$ a ternary code.

difficulties. In Sect. 2, basic settings are described and a conventional decoding method based on the BT model is reviewed. In Sect. 3, we propose a 'ternary' Boosting algorithm and a couple of new decoding methods that substantially reduce the computational cost. In Sect. 4, performance of the proposed methods is compared with those of the existing multi-class classification methods using various benchmark problems.[2]

## 2 Settings and methods

In this section, we define multi-class classification problems according to the framework of ECOC (Sect. 2.1) and then introduce a classification (decoding) method based on the BT model (Sect. 2.2).

### 2.1 Multi-class classification in the ECOC framework

Let $\boldsymbol{x}$ be a feature (input) vector to be classified and $y \in \mathcal{Y} = \{1, \ldots, G\}$ a multi-class ($G$-class) label of the feature vector $\boldsymbol{x}$, provided that a dataset consisting of $n$ examples $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ has been given for making a multi-class classifier.

We consider the decomposition of an original $G$-class classification problem into multiple binary classification problems. Let $W \in \{+1, 0, -1\}^{J \times G}$ be a code word matrix, which is assumed to be given *a priori*, and $\boldsymbol{z}(y) = (z_1(y), \ldots, z_J(y))' \in \{+1, 0, -1\}^J$ the $y$-th column vector of $W$, where $'$ denotes a transpose. The vector $\boldsymbol{z}(y)$ equivalently represents the multi-class label $y$, and we call it the 'code word' of $y$. For a given code word matrix, $W$, the original multi-class classification problem is decomposed into $J$ binary classification problems and the $j$-th row of the code word matrix, $W_j$, defines a binary classification problem; if $W_{jk} = +1$, where $W_{jk}$ is the $(j, k)$ component of $W$, then the feature vector belonging to class $k$ is regarded as a positive example for the $j$-th binary classification problem, but as a negative example if $W_{jk} = -1$. Examples belonging to the $k$-th class with zero code ($W_{jk} = 0$) are not used for training of the $j$-th classifier. The $j$-th binary classifier is thus trained by examples regarded as positive and negative. After training of all binary classifiers associated with the code word matrix, the class of a new input vector is predicted by an appropriate decoding method by integrating the outputs from the trained binary classifiers.

Let a binary classification problem designated by the $j$-th row of $W$ ideally discriminate between the positive class set $C_j^{+1}$ and the negative class set $C_j^{-1}$. By definition, $C_j^{+1}$ and $C_j^{-1}$ are non-empty and disjoint subsets of $\mathcal{Y} = \{1, \ldots, G\}$:

$$C_j^{+1} \cup C_j^{-1} \subset \{1, \ldots, G\}, \quad C_j^{+1} \neq \emptyset,\ C_j^{-1} \neq \emptyset,\ C_j^{+1} \cap C_j^{-1} = \emptyset. \tag{1}$$

Let $C_j^0$ be the complementary set of $C_j^{+1} \cup C_j^{-1}$. The $j$-th binary classifier is therefore trained using examples whose supervised labels are in $C_j^{+1} \cup C_j^{-1}$, while examples with labels in $C_j^0$

[2] A short version of this article has been presented as a conference paper (Takenouchi and Ishii 2008). While our fast decoder in the previous version used some approximations, we examine in the current study the properties of the improved fast decoder which is solved strictly by the quadratic programming technique. Another major difference between the current version and the short version are in the part of experiments, where properties and performance of the proposed method are intensively investigated to clarify the advantage and disadvantage of the proposed methods. Moreover, we compared the proposed methods with a previously proposed ECOC-based multi-class classification method (Angulo et al. 2006) and the state-of-the-art Boosting methods, which further show the properties of our methods.

are not used. As a constituent binary classifier, we can employ any one, *e.g.*, SVM (Vapnik 1995) and AdaBoost (Freund and Schapire 1997). The discriminant function of the $j$-th trained binary classifier is denoted by $f_j(\boldsymbol{x}) \in R$.

## 2.2 Bradley-Terry model-based method

A BT model-based decoding method integrates outputs from multiple binary classifiers into multi-class classification. Each binary classifier is required to output a probabilistic guess rather than a binary one. One well-established way to output a probabilistic guess is to employ the logistic model, which transforms a discriminant function value to a probabilistic value (Platt 1999). For an input $\boldsymbol{x}$ whose discriminant function value by the $j$-th binary classifier is $f_j(\boldsymbol{x}) \in R$, the probabilistic guess for binary assignment, $\boldsymbol{q}_j(\boldsymbol{x})$, is given by

$$\boldsymbol{q}_j(\boldsymbol{x}) = (q_j^{+1}(\boldsymbol{x}), q_j^{-1}(\boldsymbol{x}))', \quad q_j^{-1}(\boldsymbol{x}) = 1 - q_j^{+1}(\boldsymbol{x}),$$
$$q_j^{+1}(\boldsymbol{x}) = \frac{1}{1 + \exp(-a_j + b_j f_j(\boldsymbol{x}))}, \tag{2}$$

where $a_j$ and $b_j$ are regression parameters; by maximizing the log-likelihood

$$\sum_{i=1}^{n} \sum_{k \in \{+1,-1\}} \mathrm{I}(y_i \in C_j^k) \log q_j^k(\boldsymbol{x}_i), \tag{3}$$

maximum likelihood (ML) estimation of the parameters, $\hat{a}_j$ and $\hat{b}_j$, is performed. The function $\mathrm{I}(R)$ takes 1 if a conditional expression $R$ is true and 0 otherwise. After the parameter estimation, the probabilistic guess is obtained as $\hat{\boldsymbol{q}}_j(\boldsymbol{x})$.

For a new input vector $\boldsymbol{x}$, let

$$\boldsymbol{p}(\boldsymbol{x}) = (p_1(\boldsymbol{x}), \ldots, p_G(\boldsymbol{x}))', \quad \sum_{y \in \mathcal{Y}} p_y(\boldsymbol{x}) = 1$$

be the multinomial assignment vector, each of whose components represents the conditional probability of $y$ given $\boldsymbol{x}$. Given any assignment vector $\boldsymbol{p}(\boldsymbol{x})$, the binary assignment probability into either $C_j^{+1}$ or $C_j^{-1}$ is written as

$$\boldsymbol{\pi}_j(\boldsymbol{x}) = (\pi_j^{+1}(\boldsymbol{x}), \pi_j^{-1}(\boldsymbol{x}))', \quad \pi_j^{-1}(\boldsymbol{x}) = 1 - \pi_j^{+1}(\boldsymbol{x}), \tag{4}$$
$$\pi_j^{+1}(\boldsymbol{x}) = \frac{\sum_{y \in C_j^{+1}} p_y(\boldsymbol{x})}{\sum_{y \in C_j^{+1} \cup C_j^{-1}} p_y(\boldsymbol{x})}. \tag{5}$$

We assume that $\hat{\boldsymbol{q}}_j(\boldsymbol{x})$ is a noisy observation of $\boldsymbol{\pi}_j(\boldsymbol{x})$. Therefore, by approximating $\hat{\boldsymbol{q}}_j(\boldsymbol{x})$ by $\boldsymbol{\pi}_j(\boldsymbol{x})$, which is dependent on $p(\boldsymbol{x})$, for all $j$ in some sense, the assignment vector $\boldsymbol{p}(\boldsymbol{x})$ can be estimated, which corresponds to the multi-class classification. In the existing studies (Hastie et al. 2001; Zadrozny 2001), the weighted sum of the Kullback-Leibler (KL) divergence between $\hat{\boldsymbol{q}}_j(\boldsymbol{x})$ and $\boldsymbol{\pi}_j(\boldsymbol{x})$ was employed:

$$D(\boldsymbol{p}) = \sum_{j=1}^{J} w_j \sum_{k \in \{+1,-1\}} \hat{q}_j^k(\boldsymbol{x}) \log \frac{\hat{q}_j^k(\boldsymbol{x})}{\pi_j^k(\boldsymbol{x})}, \tag{6}$$

where $w_j$ is the weight for the $j$-th classification problem. $w_j$ can be set simply to 1 or to the number of examples $n_j$ in the set $C_j^{+1} \cup C_j^{-1}$. Alternatively, the weight $w_j$ can be estimated such that a pre-defined utility of the target multi-class classification is maximized (Yukinawa et al. 2008). By minimizing (6), the hidden assignment vector is estimated as

$$\hat{\boldsymbol{p}}(\boldsymbol{x}) = \operatorname*{argmin}_{\boldsymbol{p}} D(\boldsymbol{p}) \quad \text{subject to} \quad p_y(\boldsymbol{x}) \geq 0 \quad (y = 1, \dots, G), \ \sum_{y \in \mathcal{Y}} p_y(\boldsymbol{x}) = 1, \tag{7}$$

and the multi-class label is predicted as corresponding to the largest component of $\hat{\boldsymbol{p}}(\boldsymbol{x})$.

Note that the optimization problem (7) is usually non-convex and then difficult to solve. Since such a non-convex problem should be solved for each input $\boldsymbol{x}$, moreover, the computational cost linearly increases as the number of input vectors to be classified increases. To reduce the computational cost, an iterative algorithm was formerly presented in Hastie et al. (2001).

## 3 Classification based on ternary BT model

Consider that the $j$-th row of the code word matrix $W$ includes some zero codes. If a new input $\boldsymbol{x}$ to be classified in fact belongs to the set $C_j^0$, the class membership probability estimated by the $j$-th classifier does not have any information because the code associated with $\boldsymbol{x}$ is 0 rather than $+1$ or $-1$, and then the relation between $\boldsymbol{\pi}_j(\boldsymbol{x})$ and $\hat{\boldsymbol{q}}_j(\boldsymbol{x})$ is obfuscated. In other words, output from the $j$-th binary classifier disturbs the estimation of the assignment vector $\boldsymbol{p}(\boldsymbol{x})$ in the decoding process. To deal with this problem, one possible idea is to re-define $\boldsymbol{\pi}_j(\boldsymbol{x})$ and $\hat{\boldsymbol{q}}_j(\boldsymbol{x})$ as ternary vectors, so as to categorize the input $\boldsymbol{x}$ into either $C_j^{+1}$, $C_j^{-1}$ or $C_j^0$. For an appropriate estimation of the ternary probabilistic guess, three-class classifiers are essential. The use of a three-class classifier makes the encoding and decoding based on the general ECOC framework (*i.e.*, allowing 0 codes) consistent with each other. To achieve this in a computationally efficient manner, in this study, we present a 'ternary' AdaBoost algorithm described in Sect. 3.1. In Sect. 3.2, we propose a 'ternary' BT model-based decoding method. In Sect. 3.3, a couple of schemes for further reducing the computational cost are presented.

### 3.1 Ternary AdaBoost algorithm

In this subsection, we present an AdaBoost-type algorithm by modifying the loss function of the original AdaBoost into one dealing with the category of 'don't care' (*i.e.* code 0). Let a training dataset contain $N$ examples, $\{\boldsymbol{x}_i, \xi_i\}_{i=1}^N$, where $\xi_i \in \{+1, 0, -1\}$ is the ternary code of the $i$-th example (feature vector). Let $N_1$ be the number of examples coded $+1$ or $-1$, and $N_2$ that of examples coded 0. According to the existing BT model-based method, examples coded $+1$ or $-1$ are used for training of the binary classifier and the corresponding probabilistic guess, but examples coded 0 are just ignored. To obtain an appropriate ternary probabilistic guess, the information associated with examples coded 0 should be incorporated into the binary classifier, in particular, AdaBoost. For this purpose, we put a constraint due to the ternary nature of examples on AdaBoost's discriminant function $F(\boldsymbol{x})$: the absolute value of the discriminant function for the examples coded 0 is constrained to be small, while the absolute value for the examples coded $\pm 1$ is maximized as in the original AdaBoost. Such a constraint is achieved by the following loss function:

$$L(F) = \sum_{i=1}^{N} U(F(\boldsymbol{x}_i); \xi_i), \tag{8}$$

where

$$U(z,\xi)=\mathrm{I}(\xi\neq 0)\exp(-z\xi)+\mathrm{I}(\xi=0)\frac{\lambda}{2}(\exp(z)+\exp(-z)), \tag{9}$$

and $0\le\lambda\le 1$ is a hyper-parameter that controls the balance between the cost from code $\pm1$ and that from code 0. Note that the function (9) corresponds to the cost function of the conventional AdaBoost when $\lambda=0$. A ternary AdaBoost algorithm can be obtained so as to sequentially minimize the loss function (8) using a set of weak classification machines, $\mathcal{F}$, each of which is a weak binary classifier that outputs $+1$ or $-1$. When the discriminant function $F_{t-1}$ is given, the function is updated as

$$F_t(\boldsymbol{x})=F_{t-1}(\boldsymbol{x})+\alpha_t f_t(\boldsymbol{x}), \tag{10}$$

where $f_t\in\mathcal{F}$ is a weak machine selected as to minimize

$$\varepsilon_t(f)=\frac{\sum_{i:\xi_i\neq0}\exp(-F_{t-1}(\boldsymbol{x}_i)\xi_i)\,\mathrm{I}(f(\boldsymbol{x}_i)\neq\xi_i)+\frac{\lambda}{2}\sum_{i:\xi_i=0}\exp(F_{t-1}(\boldsymbol{x}_i)f(\boldsymbol{x}_i))}{\sum_{i=1}^{n}U(F_{t-1}(\boldsymbol{x}_i);\xi_i)}, \tag{11}$$

and $\alpha_t=\frac{1}{2}\log\frac{1-\varepsilon_t(f_t)}{\varepsilon_t(f_t)}$ is a coefficient of $f_t$.

If we set the hyper-parameter $\lambda$ to 0, the above algorithm is reduced to the original AdaBoost training algorithm. In the proposed algorithm, the quantity $\varepsilon_t(f)$ becomes small when the weak classifier $f(\boldsymbol{x})$ correctly predicts the code for an $\boldsymbol{x}$ with code $\pm1$, or the sign of $f(\boldsymbol{x})$ is different from that of the previous discriminant function $F_{t-1}(\boldsymbol{x})$ for an $\boldsymbol{x}$ with code 0. This latter character of $\varepsilon_t(f)$ allows the updated discriminant function $F_t(\boldsymbol{x})$ to output a small value for the $\boldsymbol{x}$ with code 0. This modification of the AdaBoost algorithm is the key to improving the ternary BT model-based decoding, which is described in the next subsection.

Note that the second term of (9) can be regarded as a regularization term defined by only examples with code 0 (Dekel et al. 2005), and the hyper-parameter $\lambda$ controls the degree of emphasis on code 0. Then the function $F_t(\boldsymbol{x})$ gets to ignore information of code $\pm1$ when $\lambda$ is set to a large value. When the number of examples with code 0 is large compared with these coded $\pm1$, the effect of the regularization term also becomes large.

Here, we define the risk as the expectation of the loss function (8). When we know the joint distribution $p(\boldsymbol{x})p(\xi|\boldsymbol{x})$ of a pair $(\boldsymbol{x},\xi)$, implying that the sample size $n$ goes to infinite, the minimizer of the risk has a kind of consistency as the binary classifier.

**Proposition 1** *The population minimizer $F^*$ of the risk is given by*

$$\begin{aligned}F^*(\boldsymbol{x})&\equiv\operatorname*{argmin}_F \mathrm{E}[L(F)]\\&\equiv\operatorname*{argmin}_F\int\sum_{\xi\in\{+1,0,-1\}}p(\boldsymbol{x})p(\xi|\boldsymbol{x})U(F(\boldsymbol{x}),\xi)\mathrm{d}\boldsymbol{x}\\&=\frac{1}{2}\log\frac{p(+1|\boldsymbol{x})+\frac{\lambda}{2}p(0|\boldsymbol{x})}{p(-1|\boldsymbol{x})+\frac{\lambda}{2}p(0|\boldsymbol{x})},\end{aligned} \tag{12}$$

*where $p(\xi|\boldsymbol{x})$ denotes the conditional distribution of $\xi$ given $\boldsymbol{x}$.*

Equation (12) can be obtained by a straightforward variational calculation as in Murata et al. (2004). From (12), we see that the decision boundary between classes $+1$ and $-1$

associated with $F^*$ does not depend on the value of $\lambda$, which is a preferable character of our ternary classifier.

**Proposition 2** *For any* $\boldsymbol{x}$ *in* $\{\boldsymbol{x}|0 = \operatorname{argmax}_{\xi\in\{+1,0,-1\}} p(\xi|\boldsymbol{x})\}$, *where the Bayes optimal decision* (*classification*) *becomes* 0, $|F^*(\boldsymbol{x})| \le \frac{1}{2}\log\frac{\lambda+2}{\lambda}$ *holds.*

*Proof* The function $F^*(\boldsymbol{x})$ is a monotonically increasing function with respect to $p(+1|\boldsymbol{x})$ and a monotonically decreasing function with respect to $p(-1|\boldsymbol{x})$. Because we have assumed $0 = \operatorname{argmin}_{\xi\in\{+1,0,-1\}} p(\xi|\boldsymbol{x})$ for $\boldsymbol{x}$, then for any $p(0|\boldsymbol{x})$, we observe that

$$F^*(\boldsymbol{x}) \le \frac{1}{2}\log\frac{p(0|\boldsymbol{x}) + \frac{\lambda}{2}p(0|\boldsymbol{x})}{0 + \frac{\lambda}{2}p(0|\boldsymbol{x})} = \frac{1}{2}\log\frac{\lambda+2}{\lambda}, \tag{13}$$

$$F^*(\boldsymbol{x}) \ge \frac{1}{2}\log\frac{0 + \frac{\lambda}{2}p(0|\boldsymbol{x})}{p(0|\boldsymbol{x}) + \frac{\lambda}{2}p(0|\boldsymbol{x})} = -\frac{1}{2}\log\frac{\lambda+2}{\lambda}, \tag{14}$$

showing

$$|F^*(\boldsymbol{x})| \le \frac{1}{2}\log\frac{\lambda+2}{\lambda}. \tag{15}$$

□

Note that the converse is not necessarily the case, which means that there may exist an $\boldsymbol{x}$ satisfying $|F^*(\boldsymbol{x})| \le \frac{1}{2}\log\frac{\lambda+2}{\lambda}$ but not being in $\{\boldsymbol{x}|0 = \operatorname{argmin}_{\xi\in\{+1,0,-1\}} p(\xi|\boldsymbol{x})\}$. We here stress that this ternary AdaBoost algorithm outputs nothing but a binary classifier, but its discriminant function incorporates the information that there are three classes, positive, negative and 'don't care'. A similar attempt for SVMs (tri-class SVM) was made in Angulo et al. (2006). The tri-class SVM is defined with an $(n-n_0)n_0 \times (n-n_0)n_0$ Gram matrix where $n_0$ is the number of examples with code 0, and then may require large memory: $\mathcal{O}(n^4)$ in the worst case in which $n_0$ is around $n/2$. In such a situation, the tri-class SVM requires much more computational cost than the ordinary SVM, and then needs an approximation for problems with large $n$ (and $n_0$) values, while our ternary AdaBoost can be easily employed even for large-scale problems. When employing the tri-class SVM for multi-class classification, integration of binary classifiers into multi-class classification was performed by a simple majority vote method (Angulo et al. 2006). This majority vote method is sometimes problematic, because all binary classifiers are equally treated regardless of their performance. Moreover, we cannot make a decision when outputs of the constituent binary classifiers are split. To resolve these problems which may degrade classification performance, in the following subsection, we will propose an efficient decoding method, a combination of ternary AdaBoost and the ternary BT model-based decoding, which is suitable when the ECOC-based encoding includes 0 codes.

### 3.2 Ternary BT model-based decoding

From the discriminant function of the ternary AdaBoost, we obtain membership probabilities of three categories, 'positive' ($C_j^{+1}$), 'negative' ($C_j^{-1}$) and 'don't care' ($C_j^0$), so as to correspond respectively to the $j$-th code $z_j(y)$ being $+1$, $-1$ and 0. A logistic model can model such ternary probabilistic membership as

$$\boldsymbol{q}_j(\boldsymbol{x}) = (q_j^{+1}(\boldsymbol{x}), q_j^0(\boldsymbol{x}), q_j^{-1}(\boldsymbol{x}))' = \left(\frac{e^{a_j^{+1}+b_j^{+1}F(\boldsymbol{x})}}{Z(\boldsymbol{x})}, \frac{1}{Z(\boldsymbol{x})}, \frac{e^{a_j^{-1}+b_j^{-1}F(\boldsymbol{x})}}{Z(\boldsymbol{x})}\right)', \tag{16}$$

where $a_j^{+1}, b_j^{+1}, a_j^{-1}, b_j^{-1}$ are regression parameters and $Z(\boldsymbol{x})$ is the normalization constant. Those parameters are estimated such to maximize the following log-likelihood:

$$\sum_{i=1}^{n} \sum_{k\in\{+1,0,-1\}} \mathrm{I}(y_i \in C_j^k) \log q_j^k(\boldsymbol{x}_i). \tag{17}$$

For the multinomial assignment vector $\boldsymbol{p}(\boldsymbol{x})$, we can assume the three-class membership probabilities as

$$\boldsymbol{\pi}_j(\boldsymbol{x}) = (\pi_j^{+1}(\boldsymbol{x}), \pi_j^0(\boldsymbol{x}), \pi_j^{-1}(\boldsymbol{x}))' \tag{18}$$

$$= \left( \sum_{y\in C_j^{+1}} p_y(\boldsymbol{x}), \sum_{y\in C_j^{0}} p_y(\boldsymbol{x}), \sum_{y\in C_j^{-1}} p_y(\boldsymbol{x}) \right)'. \tag{19}$$

Note that in this 'ternary' formulation, $\boldsymbol{\pi}_j(\boldsymbol{x})$ does not require any normalization term which was necessary in (5) when the $j$-th row of $W$ contains zero codes. In the special case where a row of $W$ contains no zero codes, $\boldsymbol{\pi}_j(\boldsymbol{x})$ in (19) is the same as in the original model (5).

By minimizing the weighted sum of the KL divergence between $\hat{\boldsymbol{q}}_j(\boldsymbol{x})$ and $\boldsymbol{\pi}_j(\boldsymbol{x})$,

$$D(\boldsymbol{p}) = \sum_{j=1}^{J} w_j \sum_{k\in\{+1,0,-1\}} \hat{q}_j^k(\boldsymbol{x}) \log \frac{\hat{q}_j^k(\boldsymbol{x})}{\pi_j^k(\boldsymbol{x})}, \tag{20}$$

with constraints $\sum_{y\in\mathcal{Y}} p_y(\boldsymbol{x}) = 1$ and $p_y(\boldsymbol{x}) \geq 0$ $(y \in \mathcal{Y})$, we obtain an estimator $\hat{\boldsymbol{p}}(\boldsymbol{x}) = (\hat{p}_1(\boldsymbol{x}), \ldots, \hat{p}_G(\boldsymbol{x}))'$. A multi-class label for the input $\boldsymbol{x}$ is predicted by $\mathrm{argmax}_y\, \hat{p}_y(\boldsymbol{x})$. Although we do not have a closed-form solution of the optimization problem (20), this is a convex problem so that the unique solution can be efficiently obtained by an appropriate optimization method such as a quasi-Newton method. Note that the optimization (20) is necessary for each input as in the original BT model-based decoding.

## 3.3 Reduction of computational cost

The decoding for the ternary BT models needs to minimize the (weighted) KL divergence for each input $\boldsymbol{x}$, as can be seen in (20). While (20) can be efficiently solved, the computational cost is not small when the number of input vectors whose labels are to be estimated is large. To overcome this problem, one possible idea is to simplify the optimization problem as linear (Windeatt and Ghaderi 2003). For this purpose, we consider here the following alternative objective function:

$$D_{sq}(\boldsymbol{p}) = \sum_{j=1}^{J} w_j \sum_{k\in\{+1,0,-1\}} (\pi_j^k(\boldsymbol{x}) - q_j^k(\boldsymbol{x}))^2 \tag{21}$$

$$= (\boldsymbol{t}(\boldsymbol{x}) - \boldsymbol{R}\boldsymbol{p}(\boldsymbol{x}))' \boldsymbol{V} (\boldsymbol{t}(\boldsymbol{x}) - \boldsymbol{R}\boldsymbol{p}(\boldsymbol{x})). \tag{22}$$

$\boldsymbol{t}(\boldsymbol{x})$ is a vector of $\sum_{j=1}^{J} |\boldsymbol{q}_j(\boldsymbol{x})|$ dimension, in which vectors $\boldsymbol{q}_1(\boldsymbol{x}), \ldots, \boldsymbol{q}_J(\boldsymbol{x})$ are aligned as

$$\boldsymbol{t}(\boldsymbol{x}) = (\boldsymbol{q}_1(\boldsymbol{x})', \boldsymbol{q}_2(\boldsymbol{x})', \ldots, \boldsymbol{q}_J(\boldsymbol{x})')'. \tag{23}$$

Here, $|\boldsymbol{s}|$ is the dimensionality of vector $\boldsymbol{s}$. $\boldsymbol{V}$ is a $|\boldsymbol{t}(\boldsymbol{x})| \times |\boldsymbol{t}(\boldsymbol{x})|$ diagonal matrix, each of whose diagonal components is the weight $w_j$ associated with $q_j(\boldsymbol{x})$. The weight $w_j$ of the $j$-th classifier is typically set to 1. $\boldsymbol{R}$ is a $|\boldsymbol{t}(\boldsymbol{x})| \times G$ matrix and its $j$-th row corresponding to $q_j^k(\boldsymbol{x})$ is written as

$$(\mathrm{I}(W_{j,1}=k), \mathrm{I}(W_{j,2}=k), \ldots, \mathrm{I}(W_{j,G}=k)). \tag{24}$$

In the existing BT model-based decoding method (Zadrozny 2001), the squared loss function like (21) does not make the problem linear. Because the definition (5) of $\boldsymbol{\pi}_j(\boldsymbol{x})$ includes the normalization term due to the zero codes, the squared loss function like (21) becomes non-quadratic with respect to $\boldsymbol{p}(\boldsymbol{x})$.

With the objective function (22), we consider the following optimization problem:

$$\text{Q1:} \quad \min_{\boldsymbol{p}} D_{sq}(\boldsymbol{p}), \quad \text{subject to} \quad p_y(\boldsymbol{x}) \geq 0 \quad (y=1,\ldots,G), \sum_{y\in\mathcal{Y}} p_y(\boldsymbol{x}) = 1. \tag{25}$$

Because this is a quadratic programming problem, it is efficiently solved. Moreover, a further simplification can be applied in order to reduce the computational cost more, in which the constraint of the positivity of $p_y(\boldsymbol{x})$ is removed as:

$$\text{Q2:} \quad \min_{\boldsymbol{p}} D_{sq}(\boldsymbol{p}), \quad \text{subject to} \quad \sum_{y\in\mathcal{Y}} p_y(\boldsymbol{x}) = 1. \tag{26}$$

This problem is explicitly solved. The Lagrangian is written as

$$\mathcal{L} = (\boldsymbol{t} - \boldsymbol{R}\boldsymbol{p}(\boldsymbol{x}))'\boldsymbol{V}(\boldsymbol{t} - \boldsymbol{R}\boldsymbol{p}(\boldsymbol{x})) - \gamma(\mathbf{1}'\boldsymbol{p}(\boldsymbol{x}) - 1), \tag{27}$$

where $\mathbf{1} = (1,\ldots,1)'$ is a $G$-dimensional all 1's vector and $\gamma$ is a Lagrange multiplier. The solution is given by

$$\hat{\boldsymbol{p}}(\boldsymbol{x}) = (\boldsymbol{R}'\boldsymbol{V}\boldsymbol{R})^{-1}\left(\boldsymbol{R}'\boldsymbol{V}\boldsymbol{t}(\boldsymbol{x}) + \frac{1 - \mathbf{1}'(\boldsymbol{R}'\boldsymbol{V}\boldsymbol{R})^{-1}\boldsymbol{R}'\boldsymbol{V}\boldsymbol{t}(\boldsymbol{x})}{\mathbf{1}'(\boldsymbol{R}'\boldsymbol{V}\boldsymbol{R})^{-1}\mathbf{1}}\mathbf{1}\right). \tag{28}$$

Here, $\boldsymbol{R}'\boldsymbol{V}\boldsymbol{R}$ is a $G \times G$ matrix, but its inverse can be calculated relatively easily because the number $G$ of classes are typically several tens at most. Moreover, calculations of the terms that do not depend on $\boldsymbol{t}$ in (28), including $(\boldsymbol{R}'\boldsymbol{V}\boldsymbol{R})^{-1}$, are necessary just once before decoding of the multi-class label of each input $\boldsymbol{x}$, while the decoder in Sect. 3.2 needs the optimization of (20) for each sample, *i.e.*, $n$ times in total.

In addition, we omit the constraint $\mathbf{1}'\boldsymbol{p}(\boldsymbol{x}) = 1$, which requires the total volume to be 1 for the probability distribution $\boldsymbol{p}(\boldsymbol{x})$. This simplification allows $\gamma = 0$ in (27), and hence we obtain the estimator

$$\hat{\boldsymbol{p}}(\boldsymbol{x}) = (\boldsymbol{R}'\boldsymbol{V}\boldsymbol{R})^{-1}\boldsymbol{R}'\boldsymbol{V}\boldsymbol{t}(\boldsymbol{x}). \tag{29}$$

In (29), the quantity $(\boldsymbol{R}'\boldsymbol{V}\boldsymbol{R})^{-1}\boldsymbol{R}'\boldsymbol{V}$ does not depend on the input $\boldsymbol{x}$, then decoding for an arbitrary input $\boldsymbol{x}$ is easy. Because these estimators (28) and (29) do not require $\boldsymbol{p}(\boldsymbol{x})$ to satisfy the definition of the probability distribution, positivity or constant volume, they are not suitable for estimating the conditional probability $p(y|\boldsymbol{x})$. In the classification task, however, we are usually interested in the class $\hat{y}$ that maximizes $\hat{p}_y(\boldsymbol{x})$ and then omission of constraints corresponding to requirements of the probabilistic distribution of $\boldsymbol{p}(\boldsymbol{x})$ are not very critical. In the next experiment section, we actually see that these estimators suffice for various multi-class classification tasks. For the estimator (29), we remark the following fact.

*Remark 1* If the weight $w_j$ is set to 1 and the set of binary classification problems defined by the code word matrix $W$ is invariant for arbitrary permutation of classes, then the estimator (29) is equivalent to the estimator (28).

For example, such code word matrices as "all pairs" and "1 vs all" (Allwein et al. 2001) satisfy the invariant condition in Remark 1. Then, there is no need to calculate (28) since the estimator (29) requires less computation cost.

## 4 Experiments

We examined performance of the proposed methods by comparing with the existing BT model-based decoding method using a synthetic dataset and real benchmark datasets. Since the proposed ternary modification is effective especially when the code word matrix contains a lot of 0 codes, we prepared such a code word matrix $W$ that contains $G$ rows randomly selected from the all pairs code matrix. Note that in each row of the all pairs code matrix, there is only one 1 code and one $-1$ code, and the others are all 0's. Although this setting includes the special case that $W$ is the all pairs code matrix, the training of binary classifiers in that case may be difficult, especially when the number of classes ($G$) is large. If we can employ full all pairs code matrix by sacrificing the efficiency to some extent, the performance of the proposed method increases more, which will be briefly examined in Appendix C. As the weak classifier of AdaBoost and ternary AdaBoost, we employed the decision stump, the simplest linear classifier. The step number $T$ of each of the two Boosting-type algorithms was determined by validation technique, in order to avoid overfitting, as follows: we randomly divided the original dataset into an 80% training subset and a 20% validation subset, and determined the step number so as to minimize the loss function for the validation subset. Throughout the experiments, $\lambda$ in ternary AdaBoost was fixed to 1, and for all the decoding methods, the weight $w_j$ for the $j$-th classifier was simply set to 1.

We compared the following combinations of binary classifier, probabilistic guess and decoder.

A: Binary probabilistic guess by AdaBoost and the existing BT model-based decoding with the KL-divergence (Zadrozny 2001).
B: Ternary probabilistic guess by AdaBoost and ternary BT model-based decoding with the KL-divergence.
C: Ternary probabilistic guess by ternary AdaBoost and the existing BT model-based decoding with the KL-divergence (Zadrozny 2001).
D: Ternary probabilistic guess by ternary AdaBoost, and ternary BT model-based decoding with the KL-divergence (decoder with (25)).
E: Ternary probabilistic guess by ternary AdaBoost, and ternary BT model-based decoding with the squared loss and quadratic programing (decoder with (26)).
F: Ternary probabilistic guess by ternary AdaBoost, and ternary BT model-based decoding with the squared loss. The positiveness of $p_y(\boldsymbol{x})$ was ignored (decoder with (28)).
G: Ternary probabilistic guess by ternary AdaBoost, and ternary BT model-based decoding with the squared loss. Constraints of the positiveness of $p_y(\boldsymbol{x})$ and the total volume $\mathbf{1}'\boldsymbol{p}(\boldsymbol{x}) = 1$ were ignored (decoder with (29)).

Method A is the well-established method (Zadrozny 2001; Yukinawa et al. 2008). By evaluating methods B and C, we independently examine the effect of ternary BT model-based decoding and ternary AdaBoost (training algorithm), respectively. Method D is the proposed

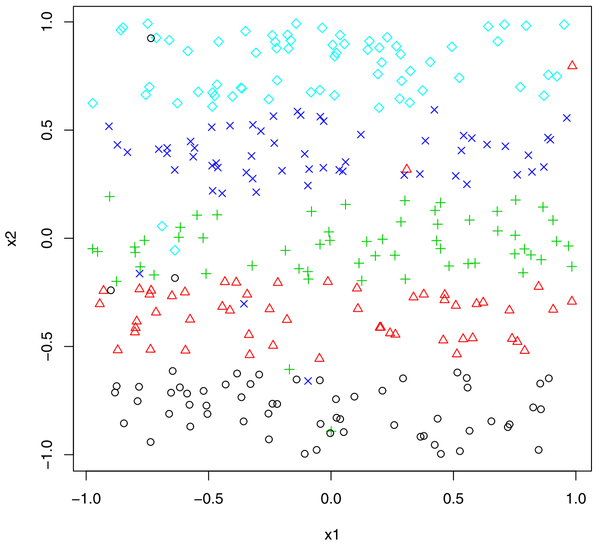


**Fig. 1** A typical synthetic dataset. Each class label is signified by a different symbol ($\circ$, $\triangle$, $+$, $\times$, $\Diamond$)

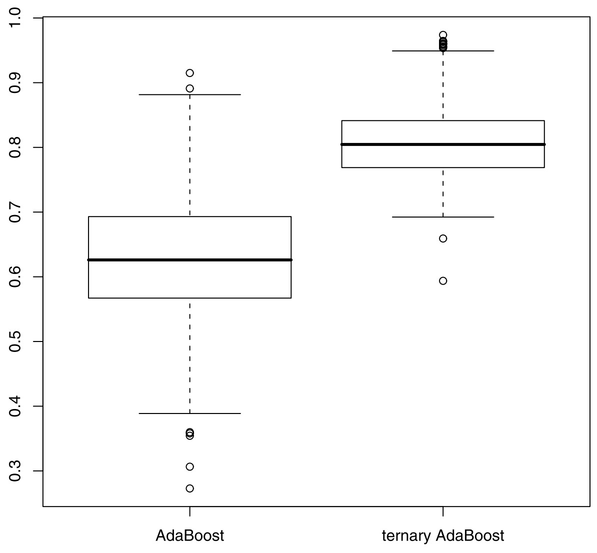
method and methods E, F and G are its simplified versions. Also as reference, comparison of the proposed methods with tri-class SVM (Angulo et al. 2006) was done with an small synthetic dataset in Appendix B.

## 4.1 Synthetic dataset

We applied the methods above, A–G, to five-class synthetic datasets each consisting of two-dimensional input feature vectors. A typical dataset is shown in Fig. 1. Note that in this simulation, only the second dimensionality $x_2$ of the feature vector $\boldsymbol{x}$ is informative for the classification task. We repeatedly generated 100 pairs of a training dataset and a test dataset each containing 300 and 3,000 samples, respectively, and examined the average performance of each method over the 100 pairs (trials). The validation subset was prepared within the training subset in the previously described manner, and the step number of Boosting-type algorithms was determined using the validation subset.

We first investigated difference between AdaBoost and ternary AdaBoost. AdaBoost and ternary AdaBoost were applied for binary classification problems defined by the all pairs code word matrix over the 100 pairs of the training datasets. In each binary problem, we calculated the Spearman's rank correlation coefficient between resultant values of discriminant function and ternary codes for examples, whose box-whisker plot is shown in Fig. 2. We observe that ternary AdaBoost attained higher correlation with the ternary code than the original AdaBoost; the difference between the two methods was statistically significant by the Wilcoxon matched-pair signed rank two-sided test, such that the $p$-value was less than $2.2 \times 10^{-16}$. Next, to observe detailed behaviors of AdaBoost and ternary AdaBoost, we examined a simple binary classification problem defined by a code word matrix $(0, 1, 0, 0, -1)$ for the synthetic dataset. Figure 3 shows the obtained discriminant functions of AdaBoost and ternary AdaBoost, which are plotted against $x_2$. Our newly proposed ternary AdaBoost could construct an appropriate discriminant function by incorporating information of examples with code 0. On the other hand, AdaBoost inconsistently predicted the code of class 3 as $+1$ or $-1$, for example; this occurred because AdaBoost emphasized discrimination between class 2 and class 5 and then failed to incorporate the information of examples coded 0.

**Fig. 2** Boxplots of the Spearman's rank correlation coefficient between ternary codes and values of the two discriminant functions constructed by AdaBoost and ternary AdaBoost



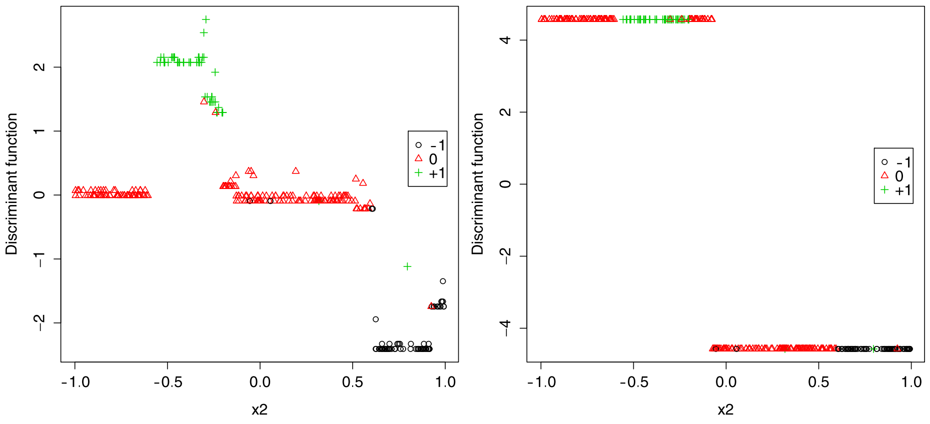


**Fig. 3** Discriminant functions obtained by ternary AdaBoost (*left panel*) and AdaBoost (*right panel*) are plotted against $x_2$. The codes of the five classes were defined as $(0, 1, 0, 0, -1)$

Figure 4 shows box-whisker plots of the training error (left panel) and the test error (right panel) over the 100 trials. These results suggest that the proposed methods, D, E, F and G, outperform the existing method (method A) and the incomplete implementations (methods B and C) are not sufficient. Additionally, one can see that performances of the decoder based on the squared loss (method E) and its simplified variants (methods F and G) are also comparable to that based on the KL-divergence (method D), while the computation time of decoders with the squared loss has been substantially reduced compared with that with the KL-divergence. Figure 5 shows box-whisker plots of the computational time for the training dataset over the 100 trials. In this figure, the upper left panel shows the computational time of processes except for decoding, which includes training of (ternary) AdaBoost and output of (ternary) probabilistic guess, and the upper right panel shows that for the decoding process, in each method. In the lower panel, we compared the computational time for all

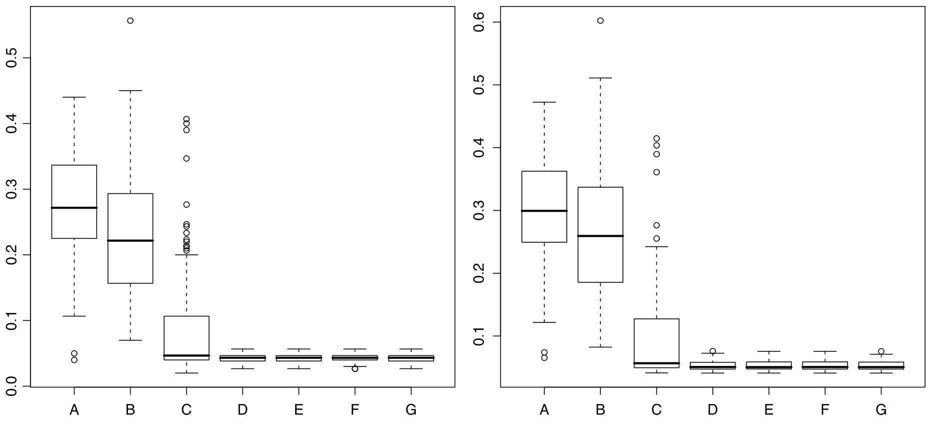


**Fig. 4** Boxplots of the training error (*left panel*) and the test error (*right panel*) over the 100 simulation trials

the processes in each method. As we can observe from the upper left panel, the consideration of the ternary code in the binary classification and the probabilistic guess requires more computational time than that without considering them, because the numbers of examples and parameters for probabilistic guess have increased by explicitly dealing with code 0. As shown in the upper right panel of Fig. 5, however, the computational time of methods E, F and G using the quadratic loss has been drastically reduced and consequently that for all the processes has also considerably improved (lower panel). Note that the optimization problems associated with methods B and D, which are defined with the KL-divergence and ternary probabilistic guess, respectively, are convex, then the computational time is reduced compared with methods A and C. Methods A, B, C and D were optimized by the quasi-Newton method registered in a library in the R language (R Development Core Team 2010) and method E was solved by a quadratic programming solver also implemented in the R language.

## 4.2 Real datasets

We examined the performance of our ternary BT model-based classification method by using real benchmark datasets, whose features are summarized in Table 1. Datasets except for the "Bioinfo" dataset were from the UCI repository (Blake and Merz 1998) and the "Bioinfo" dataset was for studying lung cancers registered at Harvard University (Bhattacharjee et al. 2001).

Since there were 8 examples with missing attributes in the Dermatology dataset, those examples were just removed; this simple heuristic method is due to the complete case analysis, whose underlying assumption of missing completely at random (MCAR) assures no bias caused by the example removal (Allison 2001).[3] Note that in our experiments, all of the methods were compared under the same condition, implying that the comparison is fair regardless of the data prepossessing including the missing entry imputation. We divided the

[3]If the MCAR assumption does not hold, any intelligent imputation method for the missing attributes would improve the classification performance. Since, however, there is no way to know whether the MCAR assumption holds or not in most real datasets, we introduced the MCAR assumption here. The detailed treatment of missing attributes is not in the scope of the current study.

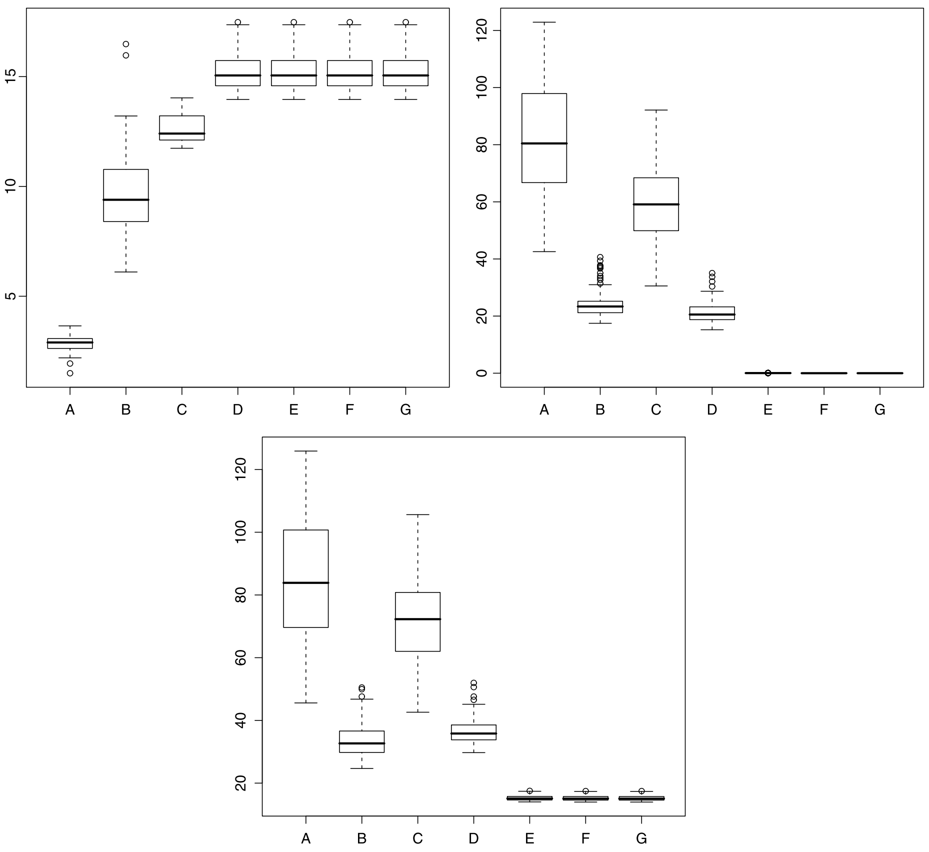


**Fig. 5** Boxplots of the computational time of the pre-decoding process (training of the binary classifier and performing of probabilistic guess) (*upper left panel*), that of the decoding process (*upper right panel*) and that of all the processes for the training dataset (*lower panel*), over the 100 simulation trials

original dataset into a training dataset and a test dataset with the ratios of 80% and 20%, respectively, and evaluated the generalization performance by using the test dataset; we repeated this procedure 100 times by changing the dataset division and observed the average performance.

Like in the simulation experiment, we subdivided the training dataset into a training subset and a validation subset with the ratios of 80% and 20%, respectively, and set the step number of the boosting algorithm so as to minimize the loss function for the validation subset.

Tables 2 and 3 show averaged performance of each method for training datasets and test datasets over the 100 trials, respectively. For each dataset, the upper row shows the mean (and the standard deviation) of error in %, and the lower row the mean of performance ratio (accuracy of each method divided by the best (bolded) one) in %. The best performance in the sense of the averaged error and standard deviation for each dataset is boldfaced. The asterisk $*$ ($**$) is attached when the error of the marked algorithm is significantly worse than that of the best method with the level of 5% (1%); we used the Wilcoxon matched-pair signed rank one-sided test. We observe that the proposed methods have drastically improved the performance over the existing BT model-based method except for few datasets.

**Table 1** Information of the real datasets

| Dataset | #Total | #Attributes | #Classes |
|---|---|---|---|
| Balance | 625 | 4 | 3 |
| Bioinfo | 203 | 2883 | 5 |
| Car eva. | 1728 | 6 | 4 |
| Dermatology | 358 | 34 | 6 |
| Glass | 214 | 9 | 6 |
| Iris | 150 | 4 | 3 |
| Sat. image | 6435 | 36 | 6 |
| Thyroid | 7200 | 21 | 3 |
| Vehicle | 846 | 18 | 4 |
| Waveform | 5000 | 21 | 3 |
| Wine | 178 | 13 | 3 |

For some datasets whose class number is small, although the existing method attained the best performance, the proposed methods also showed comparable performance. Since the median test error would be more appropriate for evaluating the expected performance of classifiers, it is shown in Table 4 of Appendix A; in terms of the median performance, our methods further outperformed the existing method. For the dataset 'Thyroid', our proposed methods were inferior to the existing method, possibly because of the unbalanced number of examples between the three classes. Actually, there are 166, 368, and 6666 examples in the three classes; when code 0 was assigned to class 3 with 6666 examples, ternary AdaBoost failed to construct an appropriate discriminant function due to the too strong regularization coming from class 3 of the large population. To deal with such an imbalanced situation, ternary AdaBoost requires an appropriate tuning of the hyper-parameter $\lambda$: we can employ the validation technique for this purpose, while it requires more computational cost being proportional to the number of binary classifiers. In most cases, the fixed value $\lambda = 1$ works well and then this setting suffices. We note that the computational cost of the decoder with the squared loss was much less than that with the KL-divergence, while the performance of the decoders with the squared loss was not degraded in spite of ignoring the constraints of the probabilistic distribution, the positiveness and the total volume. If we perform validation to appropriately tune the hyper-parameter $\lambda$, the advantage of our decoder, low computation time, may be hurt to some extent. Although we have not discussed simple but stable tuning methodologies of the hyper-parameter in this study, theoretical consideration of asymptotic statistics (Van der Vaart 1998) would be of help, which is our future work. In addition, we compared the multi-class classification performance of our methods with simple Boosting-type methods that ignore the 0 codes as reference; the results are shown in Appendix C.

As a summary of experiments, our ECOC-based multi-class decoder based on the combination of ternary AdaBoost and ternary BT model provides a fairly accurate and substantially efficient classification performance. This has been confirmed by comparison with existing ECOC-based decoders (Table 3), tri-class SVM (Fig. 7 in Appendix B) for a small-scale problem, and Boosting-type algorithms (Table 5 in Appendix C).

## 5 Conclusions

In this article, we first discussed the difficulty in applications of the BT model-based decoding to the ECOC encoding for multi-class classification tasks when the code word matrix

**Table 2** Averaged performance of each method for the real datasets in the training phase. For each dataset, the upper row shows the mean and standard deviation of error in %, and the lower row the mean of performance ratio (accuracy of each method divided by the best (bolded) one) in %

| Dataset | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| Balance | 1.426 (0.545)** | 0.474 (0.350) | 7.398 (0.612)** | **0.468 (0.480)** | **0.468 (0.480)** | **0.468 (0.480)** | **0.468 (0.480)** |
| | *99.040* | *99.996* | *93.039* | *100.000* | *100.000* | *100.000* | *100.000* |
| Bioinfo | 27.242 (17.751)** | 27.486 (4.308)** | 22.005 (19.752)** | 2.537 (2.740) | 2.557 (2.781) | 2.367 (2.499) | **2.320 (2.463)** |
| | *74.606* | *74.280* | *79.882* | *99.776* | *99.754* | *99.951* | *100.000* |
| Car eva. | 33.814 (13.030)** | 28.966 (1.115)** | 31.598 (10.844)** | **23.894 (1.275)** | 23.917 (1.219) | 23.926 (1.223) | 23.929 (1.250) |
| | *87.050* | *93.361* | *89.922* | *100.000* | *99.972* | *99.960* | *99.955* |
| Dermatology | 33.339 (11.527)** | 22.280 (8.482)** | 27.769 (9.511)** | 2.738 (1.859) | 2.601 (1.752) | 2.626 (1.805) | **2.458 (1.820)** |
| | *68.389* | *79.688* | *74.086* | *99.714* | *99.855* | *99.829* | *100.000* |
| Glass | 42.615 (12.815)** | 37.054 (9.244)** | 35.181 (12.789)** | 15.796 (6.585) | 15.510 (6.356) | 15.603 (6.306) | **15.485 (6.406)** |
| | *68.361* | *74.905* | *76.871* | *99.620* | *99.979* | *99.870* | *100.000* |
| Iris | 2.300 (2.021) | 3.183 (2.376)** | 2.533 (1.957)* | **1.975 (1.707)** | **1.975 (1.707)** | **1.975 (1.707)** | **1.975 (1.707)** |
| | *99.669* | *98.765* | *99.430* | *100.000* | *100.000* | *100.000* | *100.000* |
| Satimage | 33.104 (9.189)** | 27.977 (6.268)** | 16.372 (7.463)** | 12.973 (2.143)** | 12.719 (2.077) | 12.737 (2.116) | **12.505 (2.235)** |
| | *76.547* | *82.239* | *95.484* | *99.469* | *99.760* | *99.739* | *100.000* |
| Thyroid | **0.230 (0.273)** | 4.089 (0.642)** | 0.435 (0.122)** | 0.412 (0.172)** | 0.412 (0.172)** | 0.412 (0.172)** | 0.412 (0.172)** |
| | *100.000* | *96.131* | *99.795* | *99.818* | *99.818* | *99.818* | *99.818* |
| Vehicle | 21.189 (5.876)** | 23.435 (3.970)** | 16.675 (3.826)** | 14.444 (2.435) | 14.433 (2.480) | 14.432 (2.482) | **14.118 (2.416)** |
| | *91.819* | *89.212* | *97.037* | *99.622* | *99.632* | *99.633* | *100.000* |
| Waveform | 13.689 (0.985)** | 14.348 (0.755)** | 13.483 (0.429)** | **13.037 (0.388)** | **13.037 (0.388)** | 13.037 (0.388) | **13.037 (0.388)** |
| | *99.250* | *98.493* | *99.488* | *100.000* | *100.000* | *99.999* | *100.000* |
| Wine | 0.451 (1.018) | 1.768 (1.334)** | 0.634 (0.973) | **0.387 (0.652)** | **0.387 (0.652)** | **0.387 (0.652)** | **0.387 (0.652)** |
| | *99.939* | *98.617* | *99.754* | *100.000* | *100.000* | *100.000* | *100.000* |

**Table 3** Averaged performance of each method for the real datasets in the test phase. For each dataset, the upper row shows the mean and standard deviation of error in %, and the lower row the mean of performance ratio (accuracy of each method divided by the best (bolded) one) in %

| Dataset | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| Balance | 3.696 (1.473)** | 1.968 (1.227)** | 8.944 (2.244)** | **0.520 (0.791)** | **0.520 (0.791)** | **0.520 (0.791)** | **0.520 (0.791)** |
| | *96.812* | *98.548* | *91.535* | *100.000* | *100.000* | *100.000* | *100.000* |
| Bioinfo | 32.297 (19.137)** | 27.231 (6.853)** | 28.947 (16.954)** | 11.820 (5.573) | 11.793 (5.529) | 11.793 (5.612) | **11.766 (5.651)** |
| | *77.049* | *82.629* | *80.724* | *99.961* | *99.993* | *99.976* | *100.000* |
| Car eva. | 34.046 (13.025)** | 29.220 (2.408)** | 32.197 (10.446)** | **24.699 (2.357)** | 24.731 (2.279) | 24.728 (2.279) | 24.757 (2.288) |
| | *87.660* | *94.032* | *90.112* | *100.000* | *99.962* | *99.966* | *99.928* |
| Dermatology | 35.194 (12.399)** | 23.667 (9.282)** | 29.792 (10.773)** | 6.069 (3.485) | 5.861 (3.365) | 5.931 (3.401) | **5.750 (3.303)** |
| | *68.949* | *81.064* | *74.614* | *99.663* | *99.887* | *99.808* | *100.000* |
| Glass | 51.311 (11.926)** | 44.879 (10.599)** | 48.911 (11.573)** | 35.106 (7.581) | 35.181 (7.505) | 35.082 (7.485) | **34.908 (7.514)** |
| | *76.008* | *85.528* | *79.483* | *99.768* | *99.641* | *99.779* | *100.000* |
| Iris | 6.067 (3.651) | 7.133 (4.185)** | 6.533 (4.209) | **5.500 (3.742)** | 5.567 (3.851) | 5.533 (3.855) | **5.500 (3.742)** |
| | *99.454* | *98.327* | *98.926* | *100.000* | *99.926* | *99.962* | *100.000* |
| Satimage | 34.493 (8.843)** | 28.665 (6.268)** | 18.374 (7.029)** | 14.885 (2.240) | 14.706 (2.105) | 14.705 (2.155) | **14.490 (2.238)** |
| | *76.713* | *83.350* | *95.379* | *99.539* | *99.752* | *99.751* | *100.000* |
| Thyroid | **0.564 (0.254)** | 4.252 (0.734)** | 0.767 (0.212)** | 0.681 (0.257)** | 0.681 (0.257)** | 0.681 (0.257)** | 0.681 (0.257)** |
| | *100.000* | *96.291* | *99.796* | *99.883* | *99.883* | *99.883* | *99.883* |
| Vehicle | 29.724 (5.085)** | 29.465 (3.992)** | 27.300 (3.571) | 26.665 (3.678) | 26.788 (3.694) | 26.794 (3.708) | **26.629 (3.583)** |
| | *95.878* | *96.255* | *99.166* | *99.954* | *99.778* | *99.769* | *100.000* |
| Waveform | 16.651 (1.154)** | 16.320 (1.044) | 16.466 (1.117)* | **16.097 (1.057)** | 16.098 (1.057) | 16.098 (1.057) | 16.098 (1.057) |
| | *99.345* | *99.741* | *99.562* | *100.000* | *99.999* | *99.999* | *99.999* |
| Wine | 4.750 (3.954)* | 7.167 (4.658)** | 4.333 (3.731) | **3.750 (3.605)** | 3.778 (3.651) | 3.806 (3.654) | 3.806 (3.654) |
| | *99.030* | *96.516* | *99.418* | *100.000* | *99.970* | *99.941* | *99.941* |

includes zero codes. Because in such cases the estimated class membership probability does not have any information of the input feature vector belonging to classes with zero codes, the outputs from the trained binary classifiers would disturb the decoding of the multi-class label of the input vector to be estimated. Although this is an essential problem in the ECOC framework, it has been ignored by many ECOC-based classification studies, especially those employing BT model-based decoders. To overcome this difficulty, in this study, we proposed a ternary AdaBoost algorithm, which enables us to estimate the membership probabilities as three categories: 'positive,' 'negative' and 'don't care'. In addition, we proposed modified decoding methods using the squared loss within the framework of the 'ternary' BT model, which substantially reduced the computational cost while preserving the superior classification performance over the naive BT model-based decoding method. Based on intensive experiments using benchmark datasets, we concluded that our decoders have both of high classification performance and less computation time when the code word matrix has a fairly amount of zero codes, in comparison to the existing ECOC-based multi-class classification methods. While in the current study the weight $w_j$ in the proposed decoder was fixed, the weight can be optimized based on a pre-defined utility such as the accuracy (Yukinawa et al. 2008); the optimization with a sparse constraint such as Lasso (Tibshirani 1996) corresponds to optimizing the code word matrix, and is an interesting issue to be studied as a future work.

**Acknowledgements** This study was partially supported by Grant-in-Aid for Scientific Research on Priority Areas, "Deepening and Expansion of Statistical Mechanical Informatics," and by Grant-in-Aid for Young Scientists (B), 21700304, both from MEXT, Japan.

## Appendix A: Median performance of each method

**Table 4** Median of test errors for the real datasets

| Dataset | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| Balance | 4.000 (1.473)** | 1.600 (1.227)** | 8.800 (2.244)** | **0.000 (0.791)** | **0.000 (0.791)** | **0.000 (0.791)** | **0.000 (0.791)** |
| Bioinfo | 29.268 (19.137)** | 26.829 (6.853)** | 24.390 (16.954)** | 12.195 (5.573) | 12.195 (5.529) | **9.756 (5.612)** | 9.756 (5.651) |
| Car eva. | 33.092 (13.025)** | 29.335 (2.408)** | 27.890 (10.446)** | 24.566 (2.357) | **24.566 (2.279)** | **24.566 (2.279)** | 24.855 (2.288) |
| Dermatology | 34.722 (12.399)** | 22.222 (9.282)** | 29.167 (10.773)** | 5.556 (3.485) | 5.556 (3.365) | 5.556 (3.401) | **5.556 (3.303)** |
| Glass | 51.163 (11.926)** | 44.186 (10.599)** | 48.837 (11.573)** | 34.884 (7.581) | 34.884 (7.505) | **34.884 (7.485)** | 34.884 (7.514) |
| Iris | 6.667 (3.651) | 6.667 (4.185)** | 6.667 (4.209) | 6.667 (3.742) | 6.667 (3.851) | **5.000 (3.855)** | 6.667 (3.742) |
| Satimage | 33.800 (8.843)** | 27.506 (6.268)** | 14.763 (7.029)** | 14.530 (2.240) | 14.219 (2.105) | 14.141 (2.155) | **13.831 (2.238)** |
| Thyroid | **0.556 (0.254)** | 4.236 (0.734)** | 0.764 (0.212)** | 0.625 (0.257)** | 0.625 (0.257)** | 0.625 (0.257)** | 0.625 (0.257)** |
| Vehicle | 29.706 (5.085)** | 29.412 (3.992)** | **26.471 (3.571)** | 26.471 (3.678) | 26.471 (3.694) | 26.471 (3.708) | 26.471 (3.583) |
| Waveform | 16.750 (1.154)** | 16.200 (1.044) | 16.450 (1.117)* | **16.050 (1.057)** | **16.050 (1.057)** | **16.050 (1.057)** | **16.050 (1.057)** |
| Wine | 2.778 (3.954)* | 8.333 (4.658)** | 2.778 (3.731) | **2.778 (3.605)** | 2.778 (3.651) | 2.778 (3.654) | 2.778 (3.654) |

## Appendix B: Comparison of proposed method with tri-class SVM

A tri-class SVM that can deal with ternary code in the framework of SVMs has been proposed and employed for multi-class classification problems (Angulo et al. 2006). Since the tri-class SVM can be used for the decoder of ECOC-based multi-class classification, instead of the ternary AdaBoost we proposed, we compare the proposed classification method with the one with the tri-class SVM in this Appendix section. The tri-class SVM is defined by means of a $(n-n_0)n_0 \times (n-n_0)n_0$ Gram matrix where $n_0$ is the number of examples with code 0, and then may require a large amount of memory. Because of the limitation in computational resource, we compared the performance by using a small synthetic dataset as shown in Fig. 6. The tri-class SVM has two things to be tuned: the kernel function and the regularization term $C$ which controls the degree of emphasis on code 0 like $\lambda$ of the ternary AdaBoost. In our experiment, the kernel function was fixed to the Gaussian kernel

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2}\right),$$

and the term $C$ was selected from a set $\{2^0, \ldots, 2^{10}\}$ based on a validation technique. We repeatedly generated 100 pairs of a training dataset and a test dataset each containing 100 and 1,000 samples, respectively, and examined the average performance of each method over the 100 pairs (trials). The code word matrix was a subset of all pairs code. A validation subset was prepared within the training subset, so that hyper-parameters such as the step number of Boosting-type algorithms and the regularization term $C$ of the tri-class SVM were determined based on the validation subset.

Although in the original work (Angulo et al. 2006), the multi-class classification was performed based on a simple majority vote method, we also implemented a combination of the tri-class SVM and our BT model-based decoder to compare the performance of the decoder; we then compared the proposed methods (D, E, F and G) with the following H and I:

H: Ternary classification by tri-class SVM, and the majority vote method.

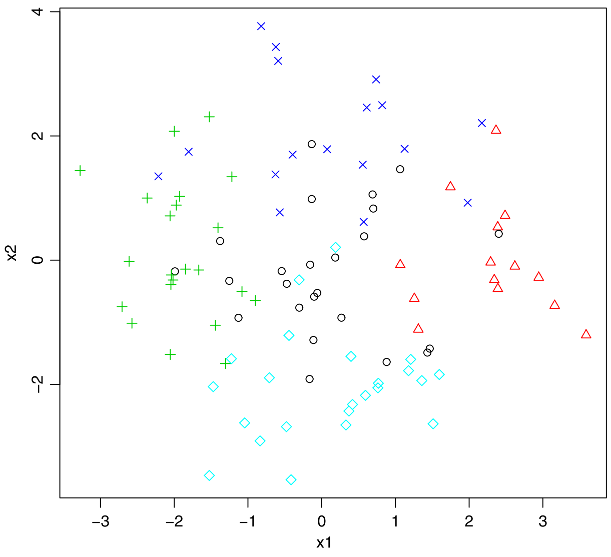
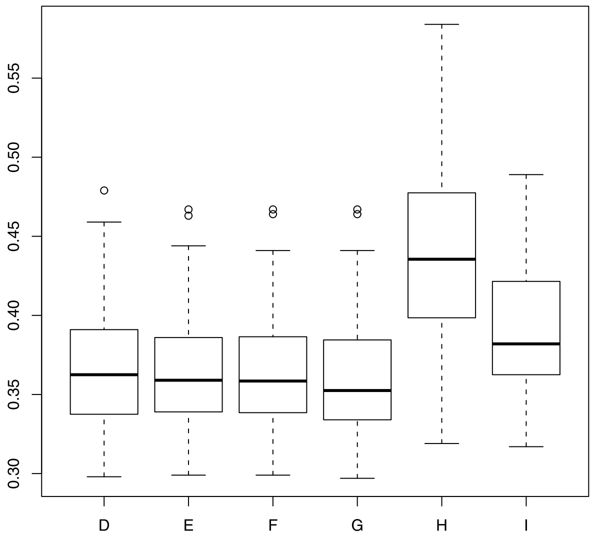


**Fig. 6** A typical synthetic dataset. Each class label is signified by a different symbol ($\circ$, $\triangle$, $+$, $\times$, $\diamondsuit$)

**Fig. 7** Boxplots of the test error over the 100 simulation trials



I: Ternary probabilistic guess by tri-class SVM, and ternary BT model-based decoding with the squared loss. Constraints of the positiveness of $p_y(\boldsymbol{x})$ and the total volume $\mathbf{1}'\boldsymbol{p}(\boldsymbol{x}) = 1$ were ignored (decoder with (29)).

Figure 7 shows box-whisker plots of the test error over the 100 trials, suggesting that the proposed methods, D, E, F and G, outperform the methods based on the tri-class SVM (methods H and I), at least in this simple problem's case. Also we observe that performance of the tri-class SVM was improved by our BT-model-based decoding method.

## Appendix C: Comparison of proposed methods with Boosting-type algorithms

The main scope of this study is to propose an efficient method which can appropriately handle the ternary code that may appear in ECOC-based multi-class classification problems. For this purpose, in Sect. 4, we have investigated detailed behaviors of the proposed methods and conventional methods in the situation where the code word matrix contains a lot of 0 codes so that 0 codes cannot be easily ignored. In this Appendix, on the other hand, we compare our methods with multi-class versions of Boosting-type classifiers, AdaBoost.M2 and AdaBoost.OC (Schapire and Singer 1999; Schapire 1997) just for reference. Although these Boosting-type methods also decompose the original multi-class problem into multiple binary problems and adaptively combine binary classifiers, its decomposition is different from that in our methods; actually, the Boosting-type methods does not use 0 codes. We applied the two Boosting-type algorithms to the same datasets described in Sect. 4.2. Averaged performance of the Boosting-type algorithms is shown in the first two column of Table 5, where we can see the Boosting-type algorithms can provide better results than those by the proposed method (Table 3) except for some datasets. This is because the number $J$ of binary classifiers used by the proposed methods was restricted to $G$ in the experiments of Sect. 4.2, to know the basic performance of our methods by comparing the existing ECOC-based decoders which ignore zero codes. Although this restriction was preferable in the point of efficiency, it in fact avoided full performance of the proposed methods. Actually,

**Table 5** Averaged performance of AdaBoost.M2, AdaBoost.OC, and the proposed methods with the all pairs code for the real datasets in the test phase. For each dataset, the upper row shows the mean and standard deviation of error in %, and the lower row the mean of performance ratio (accuracy of each method divided by the best (bolded) one) in %

| Dataset | AdaBoost.M2 | AdaBoost.OC | D | E | F | G |
|---|---|---|---|---|---|---|
| Balance | 6.696 (2.597)** | 7.208 (2.592)** | **0.520 (0.791)** | **0.520 (0.791)** | **0.520 (0.791)** | **0.520 (0.791)** |
| | *93.794* | *93.277* | *100.000* | *100.000* | *100.000* | *100.000* |
| Bioinfo | 9.584 (4.691) | 10.002 (5.591) | **9.354 (5.307)** | 9.440 (5.293) | 9.440 (5.293) | 9.440 (5.293) |
| | *99.794* | *99.562* | *100.000* | *99.907* | *99.907* | *99.907* |
| Car eva. | 25.994 (2.373)** | 25.480 (1.864)** | **24.191 (2.163)** | 24.260 (2.167) | 24.257 (2.170) | 24.257 (2.170) |
| | *97.643* | *98.339* | *100.000* | *99.909* | *99.913* | *99.913* |
| Dermatology | 4.403 (2.678) | 4.625 (2.928)* | **3.792 (2.384)** | 3.889 (2.393) | 3.875 (2.355) | 3.889 (2.377) |
| | *99.389* | *99.167* | *100.000* | *99.900* | *99.915* | *99.900* |
| Glass | 31.767 (6.856) | 32.395 (6.840) | 31.069 (6.075) | **30.834 (6.088)** | **30.834 (6.088)** | **30.834 (6.088)** |
| | *99.266* | *98.325* | *99.673* | *100.000* | *100.000* | *100.000* |
| Iris | 6.233 (3.566) | 6.333 (3.715)* | **5.500 (3.742)** | 5.567 (3.851) | 5.533 (3.826) | 5.533 (3.767) |
| | *99.265* | *99.156* | *100.000* | *99.926* | *99.962* | *99.964* |
| Satimage | 12.841 (0.893)** | 12.188 (0.972) | **12.148 (0.911)** | 12.182 (0.915) | 12.182 (0.915) | 12.182 (0.915) |
| | *99.214* | *99.958* | *100.000* | *99.962* | *99.962* | *99.962* |
| Thyroid | **0.539 (0.209)** | 0.549 (0.205) | 0.681 (0.257)** | 0.681 (0.257)** | 0.681 (0.257)** | 0.681 (0.257)** |
| | *100.000* | *99.990* | *99.858* | *99.858* | *99.858* | *99.858* |
| Vehicle | 26.265 (3.419)** | 25.529 (3.555) | **25.106 (3.207)** | 25.182 (3.179) | 25.182 (3.179) | 25.182 (3.179) |
| | *98.544* | *99.511* | *100.000* | *99.902* | *99.902* | *99.902* |
| Waveform | 15.539 (1.038) | **15.414 (1.030)** | 16.097 (1.057)** | 16.098 (1.057)** | 16.097 (1.057)** | 16.098 (1.057)** |
| | *99.856* | *100.000* | *99.197* | *99.196* | *99.197* | *99.196* |
| Wine | 5.972 (4.403)** | 6.361 (4.856)** | 3.806 (3.654) | **3.778 (3.608)** | 3.778 (3.651) | **3.778 (3.608)** |
| | *97.801* | *97.369* | *99.970* | *100.000* | *99.999* | *100.000* |

when employing the all pairs (1 vs 1) code matrix, the proposed methods outperform the Boosting-type methods in many datasets (last four columns in Table 3), with sacrificing the efficiency to some extent. However, because the proposed methods are efficient by themselves, even with the all pairs code matrix, their computation time is comparable to that of the two Boosting-type algorithms.

## References

Allison, P. D. (2001). *Missing data*. Thousand Oaks: Sage.

Allwein, E. L., Schapire, R. E., & Singer, Y. (2001). Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, *1*, 113–141.

Angulo, C., & Català, A. (2000). K-svcr. a multi-class support vector machine. In *ECML '00: Proceedings of the 11th European conference on machine learning* (pp. 31–38), London, UK. Berlin: Springer. ISBN 3-540-67602-3.

Angulo, C., Ruiz, F. J., González, L., & Ortega, J. A. (2006). Multi-classification by using tri-class SVM. *Neural Processing Letters*, *23*(1), 89–101. ISSN 1370-4621. doi:10.1007/s11063-005-3500-3.

Bhattacharjee, A., Richards, W. G., Staunton, J., Li, C., Monti, S., Vasa, P., Ladd, C., Beheshti, J., Bueno, R., Gillette, M., Loda, M., Weber, G., Mark, E. J., Lander, E. S., Wong, W., Johnson, B. E., Golub, T. R., Sugarbaker, D. J., & Meyerson, M. (2001). Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses. *Proceedings of the National Academy of Sciences of the United States of America*, *98*(24), 13790–13795.

Blake, C. L., & Merz, C. J. (1998). *UCI repository of machine learning databases*. University of California, Department of Information and Computer Science.

Bradley, R. A., & Terry, M. E. (1952). Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, *39*(3/4), 324–345.

Crammer, K., & Singer, K. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, *2*, 265–292.

Cutzu, F. (2003). Polychotomous classification with pairwise classifiers: a new voting principle. In T. Windeatt & F. Roli (Eds.), *Lecture notes in computer science: Vol. 2709. Multiple classifier systems* (pp. 115–124). Berlin: Springer. ISBN 3-540-40369-8.

Dekel, O., Shalev-Shwartz, S., & Singer, Y. (2005). Smooth epsilon-insensitive regression by loss symmetrization. *Journal of Machine Learning Research*, *6*(1), 711–741.

Dietterich, T. G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *The Journal of Artificial Intelligence Research*, *2*, 263–286.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.

Hastie, T., & Tibshirani, R. (1998). Classification by pairwise coupling. *Annals of Statistics*, *26*, 451–471.

Hastie, T., Tibishirani, R., & Friedman, J. (2001). *The elements of statistical learning*. New York: Springer.

Moreira, M., & Mayoraz, E. (1998). Improved pairwise coupling classification with correcting classifiers. In *European conference on machine learning* (pp. 160–171). URL citeseer.ist.psu.edu/moreira97improved.html.

Murata, N., Takenouchi, T., Kanamori, T., & Eguchi, S. (2004). Information geometry of $U$-boost and Bregman divergence. *Neural Computation*, *16*(7), 1437–1481.

Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers, 10*(3).

R Development Core Team (2010). *R: A language and environment for statistical computing*. Vienna: R Foundation for Statistical Computing. URL http://www.R-project.org. ISBN 3-900051-07-0.

Schapire, R. E. (1997). Using output codes to boost multiclass learning problems. In *Machine learning: Proceedings of the fourteenth international conference* (pp. 313–321). San Mateo: Morgan Kaufmann.

Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, *37*(3), 297–336.

Takenouchi, T., & Ishii, S. (2009). A multi-class classification method based on decoding of binary classifiers. *Neural Computation*, *21*(7), 2049–2081.

Takenouchi, T., & Ishii, S. (2008). Ternary Bradley-Terry model-based decoding for multi-class classification. In *IEEE workshop on machine learning for signal processing, 2008* (pp. 121–126), MLSP 2008.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B. Methodological*, *58*(1), 267–288.

Van der Vaart, A. W. (1998). *Asymptotic statistics*. Cambridge: Cambridge University Press.

Vapnik, V. (1995). *The nature of statistical learning theory*. Berlin: Springer.

Weston, J., & Watkins, C. (1999). Support vector machines for multi-class pattern recognition. In *Proceedings of the seventh European symposium on artificial neural networks* (Vol. 4, p. 6).

Windeatt, T., & Ghaderi, R. (2003). Coding and decoding strategies for multi-class learning problems. *Information Fusion*, *4*(1), 11–21.

Yukinawa, N., Oba, S., Kato, K., & Ishii, S. (2008). Optimal aggregation of binary classifiers for multi-class cancer diagnosis using gene expression profiles. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *6*(2), 333–343.

Zadrozny, B. (2001). Reducing multiclass to binary by coupling probability estimates. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in neural information processing systems* (Vol. 14). Cambridge: MIT Press.