Journal of the Operations Research
Society of Japan
2004, Vol. 47, No. 3, 182-196

A ROBUST BOOSTING METHOD FOR MISLABELED DATA

Natsuki Sano Hideo Suzuki Masato Koda
University of Tsukuba

(Received May 27, 2003; Revised February 18, 2004)

Abstract We propose a new, robust boosting method by using a sigmoidal function as a loss function. In deriving the method, the stagewise additive modelling methodology is blended with the gradient descent algorithms. Based on intensive numerical experiments, we show that the proposed method is actually better than AdaBoost and other regularized method in test error rates in the case of noisy, mislabeled situation.

Keywords: Data analysis, data mining, machine learning, boosting, AdaBoost, sigmoidal function

1. Introduction

Suppose a situation in which a management must solve a decision problem depending on a number of people whose opinions differ slightly, and their experiences and personal abilities to solve the problem seem to be almost equal and, as a result, cannot draw a decisive conclusion. To resolve this situation in order to make a reasonably better decision, is it more efficient to ignore opinions of these people and relying solely on manager's decision, or to restart the discussion by changing the team and forming a new ensemble through addition of capable people?

Obviously, the first behaviour is faster but it may not lead to a better decision. On the other hand, the second one may lead to a better decision since it will provide an iterative improvement to the solution but it certainly is a slow process and takes much longer time. This is what we often encounter during decision analysis dealing with data mining. However, we can make a better decision even in the original situation by resorting to the iterative improvement method considering that a possible better solution is still the outcome of a combination of a set of slightly different opinions converging toward the genuine better one.

The situation described above is essentially a problem associated with a committee-based decision making. The aim of this paper is to develop a new robust algorithm for pattern classification problem by using an analogy to the committee-based decision making, where each individual member of the committee corresponds to a classifier. The decision here is to correctly classify data to its true class label, and we would like to develop a new robust classification method through iterative improvement of classifiers or committee members.

In view of the above objective in mind, the starting point for the present study would be an iterative improvement procedure called " boosting, " which is a way of combining the performance of many " weak " classifiers to produce a powerful " committee. "The procedure allows the designer to continue adding weak classifiers until some desired low training error has been achieved. Boosting techniques have mostly been studied in the computational learning theory literature (e.g., see Freund [3] , Freund and Schapire [4], Schapire [14]) and received increasing attention in many areas including data mining and knowledge discovery.

While boosting has evolved over the recent years, we focus on the most commonly used version of the adaptive boosting procedure, i.e.,“ AdaBoost.M1. ”[4]. A concise description of AdaBoost is given here for the two-category classification setting. We have a set of N training data pairs, $(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_N, y_N)$, where x_i denotes a vector valued feature with $y_i = -1$ or 1, as its class label or teacher signal. The total number of component classifiers is assumed to be M. Then, the output of classification model (i.e., committee) is given by a scalar function, $F(x) = \sum_{m=1}^{M} \alpha_m f_m(x)$, in which each $f_m(x)$ denotes a classifier producing values ± 1, and α_m are prescribed constants; the corresponding prediction (i.e., decision) is $\mathrm{sign}(F(x))$.

The AdaBoost procedure trains the classifiers $f_m(x)$ on weighted versions of the training sample, by giving higher weight to cases that are currently misclassified. This is done for a sequence of weighted samples, and then the final classifier is defined to be a linear superposition of the classifiers from each stage. A detailed description of AdaBoost.M1. is summarized and given in the next box, where $I(y_i \neq f_m(x_i))$ denotes the indicator function with respect to the event such that $y_i \neq f_m(x_i)$, i.e., misclassification event, and z_i denotes an input-output pair (x_i, y_i).

AdaBoost.M1. (Freund and Schapire [4])

1. Initialize the observation weights $w_1(z_i) = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1, 2, \ldots, M$ do:
   (a) Fit a classifier $f_m(x)$ to the training data using weights $w_m(z)$.
   (b) Compute $\mathrm{err}_m = \dfrac{\sum_{i=1}^{N} w_m(z_i) I(y_i \neq f_m(x_i))}{\sum_{i=1}^{N} w_m(z_i)}$.
   (c) Compute $\alpha_m = \log((1 - \mathrm{err}_m)/(\mathrm{err}_m))$.
   (d) Update weights
   $w_{m+1}(z_i) = w_m(z_i) \cdot \exp[\alpha_m \cdot I(y_i \neq f_m(x_i))],\ i = 1, 2, \ldots, N$.
   end For
3. Output $F(x) = \mathrm{sign}[\sum_{m=1}^{M} \alpha_m f_m(x)]$.

Much has been written about the success of AdaBoost in producing accurate classifiers. Many authors have explored the use of a tree-based classifier for $f_m(x)$ and demonstrated that it consistently produces significantly lower error rates than a single decision tree. In fact, Breiman [1] called AdaBoost with trees as “ the best off-the-shelf classifier in the world. ”

Interestingly, in many examples, the test error seems to consistently decrease and then level off as more classifiers are added, instead of turning into ultimate increase. It hence seems that AdaBoost is resistant to overfitting for low noise cases. However, recent studies with highly noisy patterns (e.g., Grove and Schuurmans [7], Quinlan [12], Rätsch et al. [13]) depict that it is clearly a myth that AdaBoost does not overfit since AdaBoost asymptotically concentrates on the patterns which are hardest to learn. To cope with this problem, some regularized boosting algorithms are proposed (Mason et al. [10], Rätsch et al. [13]). In this paper, we will take an iterative improvement approach to optimize classifiers to derive a new robust boosting method that is resistant against mislabeled noisy patterns.

In the next section, we present the principles of AdaBoost. We also review a technique of forward stagewise additive modelling in Section 2. Then, in Section 3, we present the sigmoidal loss function and propose the new robust boosting method. In Section 4, results of intensive numerical experiments are presented, and we detail cases with noisy, mislabeled patterns. Section 5 contains some concluding remarks.

2. AdaBoost as Forward Stagewise Additive Modelling

2.1. Forward stagewise additive modelling

Friedman et al. [6], and Hastie et al. [8] have given an interpretation of AdaBoost as a forward stagewise additive modelling. Forward stagewise additive modelling is a greedy forward stepwise approach for fitting an additive expansion as follows:

$$F_M(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m), \tag{1}$$

where $\beta_m, m = 1, 2, \ldots, M$, are expansion coefficients, and basis functions $\{b(x; \gamma_m)\}_{m=1}^{M}$ are taken to be simple functions characterized by a set of parameter vectors γ. For example, in single hidden layer neural networks, $b(x; \gamma) = \sigma(\gamma^t x)$, where $\sigma(\cdot)$ denotes the familiar logistic function, γ parameterizes a linear combination of the input features, and $\gamma^t x$ denotes the vector inner product. Such a learning network is trained with a given set of input features and output values to compute the optimal synaptic weights employing gradient descent methods (e.g., see Koda and Okano [9]).

Typically these models are fit by minimizing an appropriate loss function averaged over the training data, such as squared-error or likelihood-based loss function,

$$\min_{\{\beta_m, \gamma_m\}_{m=1}^{M}} \sum_{i=1}^{N} L(y_i, \sum_{m=1}^{M} \beta_m b(x_i; \gamma_m)), \tag{2}$$

where $L(y, F_M(x))$ denotes the loss function. To solve Equation (2) for a general class of loss functions $L(y, F(x))$ and/or basis functions $b(x; \gamma)$, computationally intensive numerical optimization techniques are required in general. The forward stegewise additive modelling is summarized and given in the following box.

Forward Stagewise Additive Modelling

1. Initialize $F_0(x) = 0$.
2. For $m = 1, 2, \ldots, M$ do:
   (a) Compute

$$(\beta_m, \gamma_m) = \arg\min_{\beta, \gamma} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \beta b(x_i; \gamma)).$$

   (b) Set $F_m(x) = F_{m-1}(x) + \beta_m b(x; \gamma_m)$.
   end For
3. Output $\mathrm{sign}(F_M(x))$.

Forward stagewise additive modelling approximates the solution to Equation (2) by sequentially adding new basis functions to the expansion without adjusting the parameters and coefficients of those that have already been added. At each iteration m, one solves for the optimal basis function $b(x; \gamma_m)$ and corresponding coefficient β_m to add to the current expansion $F_{m-1}(x)$. This produces $F_m(x)$, and the process is repeated until some desired low training error has been achieved at the M-th stage. In the boosting terminology, $b(x; \gamma)$ would be referred to as the " base learner, " and $F_M(x)$ the " committee. "

2.2. Derivation of AdaBoost

Friedman et al. [6] have demonstrated that AdaBoost.M1. is equivalent to forward stagewise additive modelling using the following exponential loss function:

$$L(y, F(x)) = \exp(-yF(x)), \tag{3}$$

where $F(x)$ denotes the classification model.

In AdaBoost, the basis function is the individual classifier $f(x) \in \{-1, 1\}$. Using the exponential loss function (3), one must solve

$$(\beta_m, f_m) = \arg\min_{\beta,f} \sum_{i=1}^{N} \exp[-y_i(F_{m-1}(x_i) + \beta f(x_i))]$$

for the optimal classifier $f_m(x)$ and corresponding coefficient β_m to be added at the m-th step. This can be expressed as

$$(\beta_m, f_m) = \arg\min_{\beta,f} \sum_{i=1}^{N} w_m(z_i) \exp(-\beta y_i f(x_i)) \tag{4}$$

with $w_m(z_i) = \exp(-y_i F_{m-1}(x_i))$, where z_i symbolically denotes the input-output pair (x_i, y_i). Since each $w_m(z_i)$ depends neither on β nor $f(x)$, it can be regarded as a weight that is applied to each observation. This weight depends on $F_{m-1}(x_i)$, and so the individual weight value changes at each iteration m.

The solution to Equation (4) can be obtained in two steps. First, for any value of $\beta > 0$, the solution to Equation (4) for $f_m(x)$ is given by

$$f_m(x) = \arg\min_{f} \sum_{i=1}^{N} w_m(z_i) I(y_i \neq f(x_i)), \tag{5}$$

which is the classifier that minimizes the weighted error rate in predicting y. This can be easily seen by expressing the criterion in Equation (4) as

$$e^{-\beta} \cdot \sum_{y_i = f(x_i)} w_m(z_i) + e^{\beta} \cdot \sum_{y_i \neq f(x_i)} w_m(z_i),$$

which in turn can be written as

$$(e^{\beta} - e^{-\beta}) \cdot \sum_{i=1}^{N} w_m(z_i) I(y_i \neq f(x_i)) + e^{-\beta} \cdot \sum_{i=1}^{N} w_m(z_i). \tag{6}$$

Plugging $f = f_m$, i.e., Equation (5), into Equation (4) and solving for the optimal value of β, one obtains

$$\beta_m = \frac{1}{2} \log \frac{1 - \mathrm{err}_m}{\mathrm{err}_m}, \tag{7}$$

where err_m is the optimal error rate defined as

$$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_m(z_i) I(y_i \neq f_m(x_i))}{\sum_{i=1}^{N} w_m(z_i)}. \tag{8}$$

Note that Equation (8) is equivalent to line 2(b) of AdaBoost.M1. algorithm. The representation of $F_m(x)$ is then updated as

$$F_m(x) = F_{m-1}(x) + \beta_m f_m(x),$$

which renews the weights for the next iteration as follows:

$$w_{m+1}(z_i) = w_m(z_i) \cdot e^{-\beta_m y_i f_m(x_i)}. \tag{9}$$

Using the fact that $-y_i f_m(x_i) = 2 \cdot I(y_i \neq f_m(x_i)) - 1$, Equation (9) becomes

$$w_{m+1}(z_i) = w_m(z_i) \cdot e^{\alpha_m I(y_i \neq f_m(x_i))} \cdot e^{-\beta_m}, \tag{10}$$

where $\alpha_m = 2\beta_m$ is the quantity defined at line 2(c) of AdaBoost.M1. The factor $e^{-\beta_m}$ in Equation (10) multiplies all weights by the same value, so it has no effect. Thus Equation (10) is equivalent to line 2(d) of AdaBoost.M1. algorithm. One can view line 2(a) of the algorithm as a method for solving the minimization involved in Equation (5). Hence we conclude that AdaBoost.M1. minimizes the exponential loss criterion (3) through a forward stagewise additive modeling approach.

3. Derivation of Boosting Method Using Sigmoidal Loss Function

3.1. Sigmoidal loss function

Friedman et al. [6] have demonstrated that the AdaBoost algorithm decreases an exponential loss function through the forward stagewise additive modelling along the lines that are presented in Section 2. Figure 1 illustrates various loss functions as a function of the margin value, $y \cdot F(x)$, including the exponential loss function. When all the class labels are not mislabeled and hence data is error-free, the result of correct classification always yields a positive margin since y and $F(x)$ both share the same sign while incorrect one yields negative margin. The loss function for Support Vector Machine is also shown in Figure 1, which is a statistical learning method to train kernel-based machines with optimal margins by mapping training data in a higher dimensional space.

Shown also in Figure 1 is the misclassification loss, $L(y, F(x)) = I(y \cdot F(x) < 0)$, where $I(y \cdot F(x) < 0)$ denotes the indicator function with respect to the occurrence of the incorrect events, i.e., $y \cdot F(x) < 0$, which gives unit penalty for negative margin values, with no penalty for positive ones (i.e., correct decisions). Note that the misclassification loss is a discontinuous step function. In this way, the decision rule becomes a judgment on a zero-one loss function.

The exponential loss function exponentially penalizes negative margin observations or incorrect decisions. At any point in the training process, the exponential criterion concentrates much more influence on observations with large negative margins. This is considered as one of the reasons why AdaBoost is not robust for noisy situation where there is misspecification of the class labels in the training data. Since the misclassification loss concentrates and uniquely influences on negative margin, it is far more robust in noisy setting where the Bayes error rate is not close to zero, which is especially the case in mislabeled situation. Here we would like to propose a loss function which takes account of limited influences from larger negative margins in a proper manner and, accordingly, robust against mislabeled noisy training data.

Mean-squared approximation errors are well-understood and used as a loss function in statistics area. Unlike the misclassification loss which considers only the misclassified

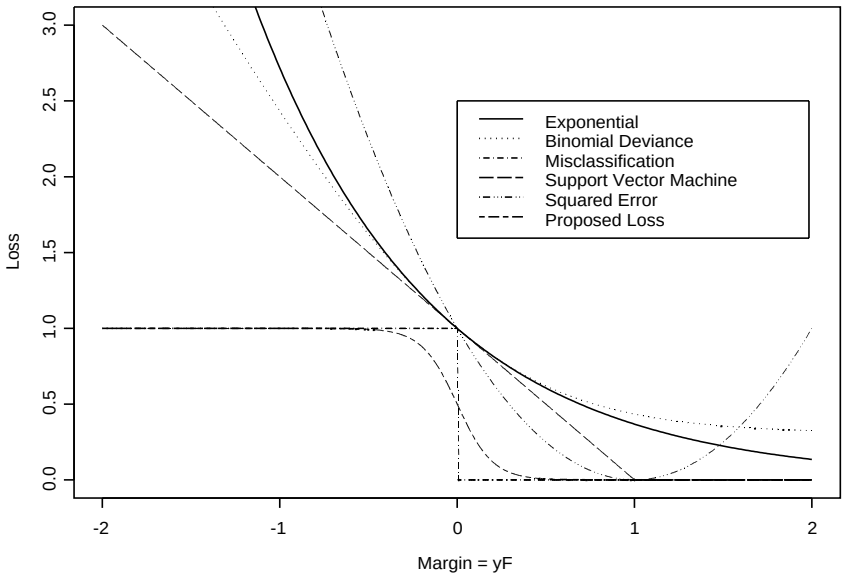


Figure 1: Loss functions for two-category classification (Following [8]).

observations, the minimum squared error (MSE) criterion takes into account the entire training samples with wide range of margin values. Hence, if MSE is adopted, the correct classification but with $y \cdot F(x) > 1$ incurs increasing loss for larger values of $|F(x)|$. This makes the squared-error a poor approximation compared to the misclassification loss and not desirable since the classification results that are " excessively " correct are also penalized as much as worst (extremely incorrect) cases. Other functions in Figure 1 can be viewed as monotone continuous approximations to the misclassification loss. Friedman et al. [6] derived " LogitBoost " based on a binomial deviance loss.

Actual misclassification loss is not continuous and therefore we propose the following continuous function to approximate the misclassification loss,

$$L(y, F(x)) = \frac{1}{1 + e^{\kappa y F(x)}}. \tag{11}$$

This is a sigmoidal function where κ denotes the appropriate positive gain. Note that the proposed loss function (11) is mirror symmetric to the familiar logistic function with respect to the vertical axis located at $y \cdot F(x) = 0$, i.e., zero margin axis. Then, an optimization of Equation (11) by using the stegewise additive modelling can be formulated as follows:

$$\begin{aligned} (\beta_m, \gamma_m) &= \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \beta b(x_i; \gamma)), \\ &= \arg\min_{\beta,\gamma} \sum_{i=1}^{N} \frac{1}{1 + \exp(\kappa y_i (F_{m-1}(x_i) + \beta b(x_i; \gamma)))}, \end{aligned} \tag{12}$$

where $b(x; \gamma)$ denotes an appropriate base learner such as neural network. It should be noted that the solution to Equation (12) is very difficult to obtain in general, since it involves simultaneous optimization with respect to the two model parameters, β and γ.

3.2. Proposed boosting algorithm

In this study, the optimization involved in Equation (12) is approximately executed by using an analogy to the one that is typically employed in numerical optimization. In general, the approximation of functions using input-output relations may be converted to a parameter optimization problem by selecting an appropriately parameterized model. However, we take a " nonparametric " approach without assuming any parameterized models and hence we apply numerical optimization procedures in function space. The readers are referred to Friedman [5] for details of the approximation techniques.

The empirical loss in using $F(x)$ to predict y on the training data is given by

$$L^N = \sum_{i=1}^{N} L(y_i, F(x_i)). \tag{13}$$

Ordinarily this empirical loss is a function of the modelling parameters, i.e., β and γ. However, without assuming such a parameterized model, we optimize Equation (13) using a novel approach. Let $\mathbf{F} = \{F(x_1), \cdots, F(x_i), \cdots, F(x_N)\}$ be considered as " parameters " to approximate the continuous function $F(x)$ at each of the N data points x_i. Then, minimization of Equation (11) can be viewed as a numerical optimization problem and hence gradient descent methods can be effectively applied. In the present stagewise additive modelling formulation, at the m-th iteration stage, the components of the gradient $\mathbf{g}_m$ evaluated at $\mathbf{F} = \mathbf{F}_{m-1}$ are given by

$$g_{im} = \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x_i)=F_{m-1}(x_i)}. \tag{14}$$

The gradient, $\mathbf{g}_m \in \mathbf{R}^N$, is defined only at the training data point x_i, whereas our ultimate goal is to generalize the output of classification model, $F(x)$, to new unknown data x not represented in the training samples. In order to fit negative gradients, $-\mathbf{g}_m$, method of least squares minimization is employed here. The final classification may be achieved by using relevant function approximation techniques through linear or nonlinear superpositions of functions defined at data points. By superpositions, however, even though the class label y is binary-valued, $F(x)$ outputs a continuous value, not ± 1. This resembles the noisy version of the Bayes discriminant analysis where the conditional mean of y takes continuous values for binary-valued y and offers a fair amount of flexibility needed for generalization capability. Hence, the proposed approach may be robust when it is applied to noisy situations with mislabeled events.

From Equation (14), we compute the components of negative gradient, $-g_{im}$, utilizing the proposed loss function (11) as follows:

$$-g_{im} = -\left[\frac{\partial}{\partial F(x_i)}\frac{1}{1+e^{\kappa y_i F(x_i)}}\right]_{F(x_i)=F_{m-1}(x_i)} = \left.\frac{\kappa y_i e^{\kappa y_i F(x_i)}}{(1+e^{\kappa y_i F(x_i)})^2}\right|_{F(x_i)=F_{m-1}(x_i)}. \tag{15}$$

Using Equation (15), method of least squares minimization can be formulated as

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{N} [g_{im} + b(x_i;\gamma)]^2, \tag{16}$$

for an appropriate base learner $b(x;\gamma)$ to fit to the negative gradient $-g_{im}$.

At data point x_i, the gradient descent algorithm yields the following adaptation rule:

$$F_m(x_i) = F_{m-1}(x_i) + \delta\hat{g}_{im} = F_{m-1}(x_i) - \beta_m g_{im}, \tag{17}$$

where $\hat{g}_{im}$ denotes the estimated negative gradient, and the " correction " term $\delta\hat{g}_{im} = -\beta_m g_{im}$ is generally a function of the input feature x_i, its class label, and the current estimate $F_{m-1}(x_i)$, and β_m (> 0) denotes the learning parameter that is used for a generalization of gradient features at the m-th stage. Note that β_m is adjusted after the entire training set is classified. Then, for unknown x, Equation (17) can be generalized as

$$F_m(x) = F_{m-1}(x) + \beta_m b(x; \gamma_m), \tag{18}$$

where $b(x; \gamma_m)$ denotes the optimal base learner obtained from Equation (16).

In view of our ultimate target $F(x) = y$, $F_0(x)$ is initialized to $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$, the mean value of y_i. This concludes the derivation of the algorithm and the proposed method can be summarized as follows.

Proposed Method

1. Initialize $F_0(x) = \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$, and specify K.
2. For $m = 1, 2, \ldots, M$ do:
   (a) $g_{im} = \left.\frac{-\kappa y_i e^{\kappa y_i F(x_i)}}{(1+e^{\kappa y_i F(x_i)})^2}\right|_{F(x_i)=F_{m-1}(x_i)}$
   (b) $\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{N} [g_{im} + b(x_i; \gamma)]^2$
   (c) $\beta_m = \frac{K}{K+m}$
   (d) $F_m(x) = F_{m-1}(x) + \beta_m b(x; \gamma_m)$
   end For
3. Output $\mathrm{sign}(F_M(x))$.

For the appropriate choice of β_m, we propose $\beta_m = K/(K+m)$, where K is a suitable integer. In our numerical experiments in Section 4, K is set to M. This setting of β_m is based on the stochastic approximation technique [2] which satisfies the conditions

$$\lim_{M\to\infty} \sum_{m=1}^{M} \beta_m = \infty, \text{ and } \lim_{M\to\infty} \sum_{m=1}^{M} \beta_m^2 < \infty. \tag{19}$$

The proposed method of β_m guarantees a convergence of F_m if an infinite sequence is applied to the gradient-type search method, such as Equation (17) in this study. It is noted that the present boosting method involves a gradient numerical search in order to generalize gradient features, and the solution provided by the method may be a local minimum and not necessarily be a global optimal solution.

4. Numerical Experiments

In this section, numerical results are presented and, especially, the robustness of the proposed method is analyzed and compared with that of AdaBoost and AdaBoost$_{Reg}$. We focus our attention to classification results for the mislabeled (i.e., noisy) case near decision boundary, the mislabeled case far from decision boundary, and correctly labeled (i.e., noiseless) case. The back-propagation neural network with single hidden layer is used as a base learner for all the three cases. We may note that a multilayer neural network has been shown to be able

to define an arbitrary decision function, with a flexible architecture in terms of the number of hidden units. For the present numerical experiments, the number of hidden units is fixed to 3.

4.1. Generation of mislabeled data

For numerical experiments, data (with 2% mislabeled case) is generated as follows:

1. Decide N, the size of training and test examples;
2. generate $2N$ training and test examples from five dimensional standard normal distribution $\mathbf{x} \sim N^5(0, I)$;
3. decide r^2, the squared-radius from the origin, for all examples;

$$r(\mathbf{x}_i)^2 = \sum_{j=1}^{5} x_{ij}^2 \ (i = 1, \ldots, 2N)$$

4. decide threshold th by the median of $r(\mathbf{x})^2$;
5. assign $y = \mathrm{sign}(F(\mathbf{x}))$, where $F(\mathbf{x}) = r(\mathbf{x})^2 - th$;
6. sort $|r(\mathbf{x}_i)^2 - th|$ $(i = 1, \ldots, N)$ corresponding to training examples by descending order;
7. sample randomly 2% from top (or below) 50% from the outcome of Step 6;
8. flip the label of sampled examples in Step 7.

In Step 5, note that $F(\mathbf{x}) = r(\mathbf{x})^2 - th$ is used as a nonlinear decision function. In the mislabeled case at far from decision boundary, samples from top 50% training examples are selected in Step 7. In the mislabeled case at near decision boundary, samples from below 50% training examples are chosen as training examples. It is noted that the mislabeled data is only contained in training examples $\mathbf{x}_i$ $(i = 1, \ldots, N)$, and the test examples $\mathbf{x}_i$ $(i = N+1, \ldots, 2N)$ are noise free. Above procedure is repeated 5 times, each time different data set is generated and the performance of boosting methods for each data set is evaluated. In Figures 2, 3, and 4, the average performance of training and test error rates is shown.

4.2. Other boosting method $-$AdaBoost$_{Reg}$

In order to verify the proposed method, the performance of the method is compared with AdaBoost and AdaBoost$_{Reg}$. AdaBoost$_{Reg}$ is the regularized boosting method proposed by Rätsch et al. [13]. They defined regularization term $\mu_m(z_i)$ to express the *influences* of a pattern on the combined classifiers f_m by

$$\mu_m(z_i) = \sum_{s=1}^{m} c_s w_s(z_i).$$

where c_s denotes the normalized weights of classifiers such that $\sum_{s=1}^{m} c_s = 1$, and z_i denotes the input-output pair (x_i, y_i). If there is mislabeled data in training data, $\mu_m(z_i)$ becomes a quantity to express *mistrust* of the corresponding pattern z_i. Since AdaBoost asymptotically concentrate on the patterns which are hardest to learn, they introduced the loss function of AdaBoost$_{Reg}$ as follows:

$$G_{Reg}(\beta^m) = \sum_{i=1}^{N} \exp\left\{ -\frac{1}{2}[\rho(z_i, \beta^m) + C|\beta^m|\mu_m(z_i)^2] \right\}, \tag{20}$$

where $\beta^m = (\beta_1, \beta_2, \cdots, \beta_m)$, and $\rho(z_i, \beta^m)$ denotes the margin for an input-output pair $z_i = (x_i, y_i)$ as follows:

$$\rho(z_i, \beta^m) = y_i F(x_i) = y_i \sum_{s=1}^{m} \beta_s f_s(x_i). \tag{21}$$

Note that the optimal value of β_m is obtained efficiently through a line search procedure (e.g., Press et al. [11]) by minimizing Equation (20), which has a unique solution because $\frac{\partial}{\partial \beta_m} G_{Reg}(\beta^m) > 0$ is satisfied for $\beta_m > 0$. AdaBoost$_{Reg}$ is summarized and given in the following box, in which C is a priori chosen constant, and $|\beta^m|$ is l_1 norm of β^m such that $|\beta^m| = \sum_{s=1}^{m} |\beta_s|$. If $C = 0$ the original AdaBoost algorithm is retrieved. The readers are reffered to Rätsch et al. [13] for details of AdaBoost$_{Reg}$.

AdaBoost$_{Reg}$ (Rätsch et al. [13])

1. Initialize the observation weights $w_1(z_i) = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1, 2, \ldots, M$ do:
   (a) Fit a classifier $f_m(x)$ to the training data using weights $w_m(z)$.
   (b) Find β_m by line search

$$\beta_m = \arg\min_{\beta_m \geq 0} \sum_{i=1}^{N} \exp\left\{ -\frac{1}{2}[\rho(z_i, \beta^m) + C|\beta^m|\mu_m(z_i)^2] \right\}.$$

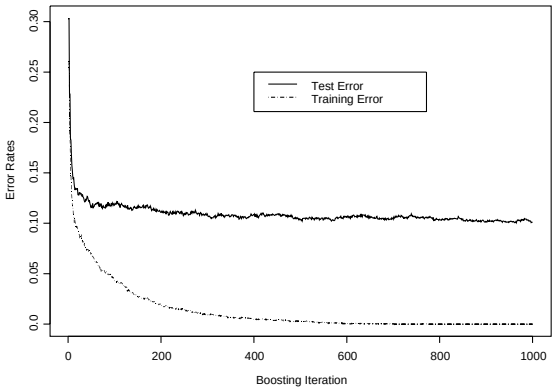
   (c) Update weights
   $w_{m+1}(z_i) = \frac{1}{Z_{m+1}} \exp\left\{ -\frac{1}{2}[\rho(z_i, \beta^m) + C|\beta^m|\mu_m(z_i)^2] \right\}$,
   $i = 1, 2, \ldots, N$, where Z_{m+1} denotes the normalization constant, such that $\sum_{i=1}^{N} w_{m+1}(z_i) = 1$.
   end For
3. Output $F(x) = \text{sign}(\sum_{m=1}^{M} \beta_m f_m(x))$.

4.3. Results of experiment

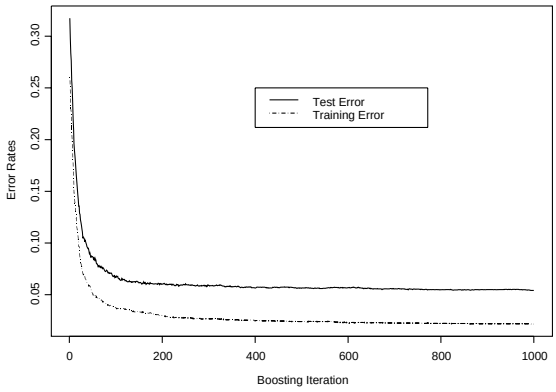
The number of training and test data are 1000 each, i.e., $N = 1000$. The independent experiments on the noiseless situation in which no mislabeled data is contained on the same data set are also conducted. Iteration number of the weak learner, which is referred to as round (number), is 1000. The gain parameter of the proposed loss function, κ in Equation (11), is set to 1, and the regularized parameter of AdaBoost$_{Reg}$, C, is set to 10000, where the value of C is determined based on preliminary experiments.

We plot in Figure 2 the learning (error minimization) curves for 2% mislabeled cases far from decision boundary. In Figure 3, the learning curves of mislabeled cases near decision boundary are shown. In Figure 4, we plot learning curves of noiseless data. In each figure, (a) shows AdaBoost training and test error rates, (b) training and test error rates of the proposed method, (c) training and test error rates of AdaBoost$_{Reg}$, and (d) comparison of test error rates of all the three boosting methods. On test error rates, we may note that the performance of the proposed method is superior to that of other two methods.

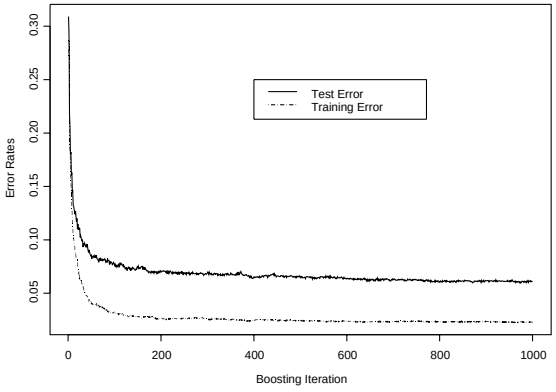
In Figure 2 (d), the test error rate of the proposed method falls into 0.054% at 1000 round, which is lower than that of other two boosting methods. In Figure 3 (d), the test error rate of the proposed method falls into 0.051% at 1000 round, which is lower than the other two boosting methods. In Figure 4 (d), the test error rate of the proposed method falls into 0.045% at 1000 round, which exhibits similar performance to the other two methods. In mislabeled cases in Figures 2 and 3, the superiority of the proposed method is remarkable. Even in noiseless case, the performance of the proposed method is equivalent to that of other two boosting methods. In summary, the present method achieved superior performance in general.



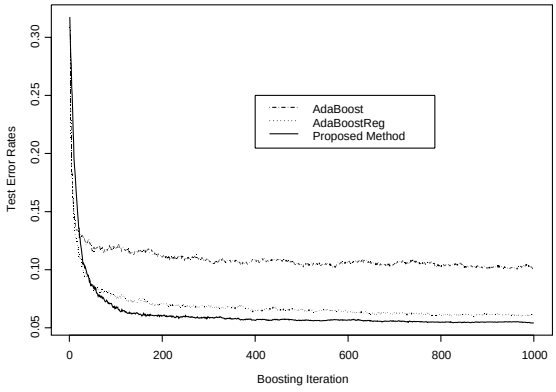
(a) AdaBoost



(b) Proposed method

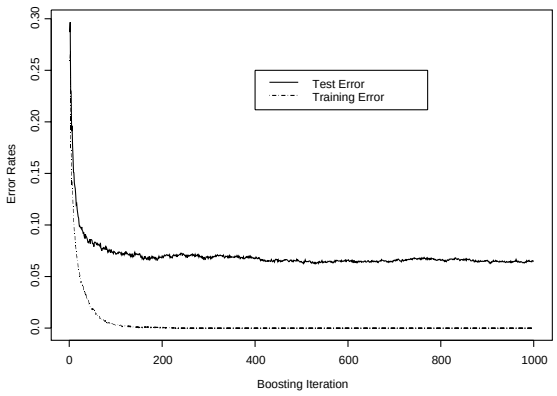


(c) AdaBoost$_{Reg}$

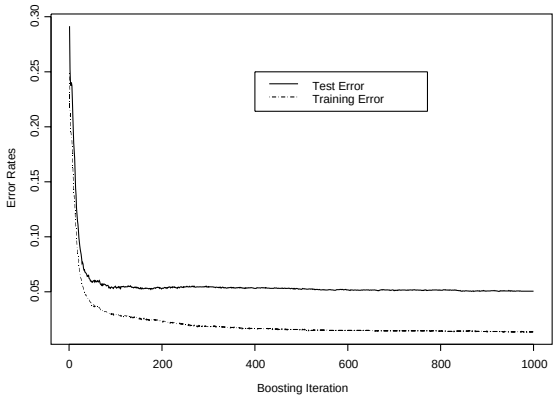


(d) Three methods

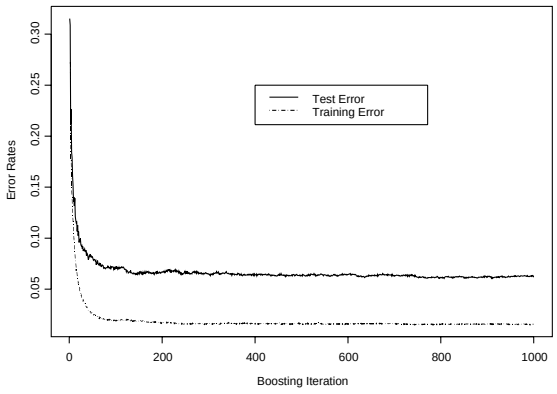
Figure 2: Comparison of three boosting methods on 2% mislabeled cases far from decision boundary. (a) Learning and test error rates of AdaBoost. (b) Learning and test error rates of proposed method. (c) Learning and test error rates of AdaBoost$_{Reg}$. (d) Test error rates of three boosting methods.



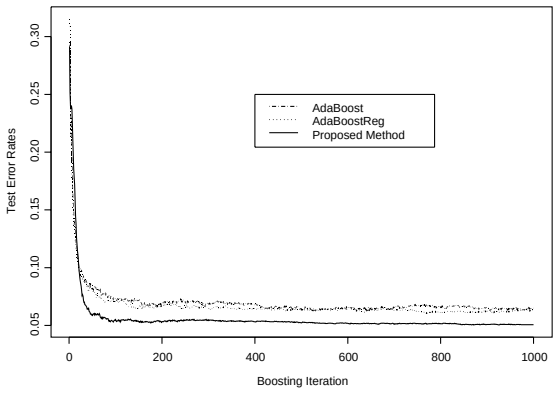
(a) AdaBoost



(b) Proposed method

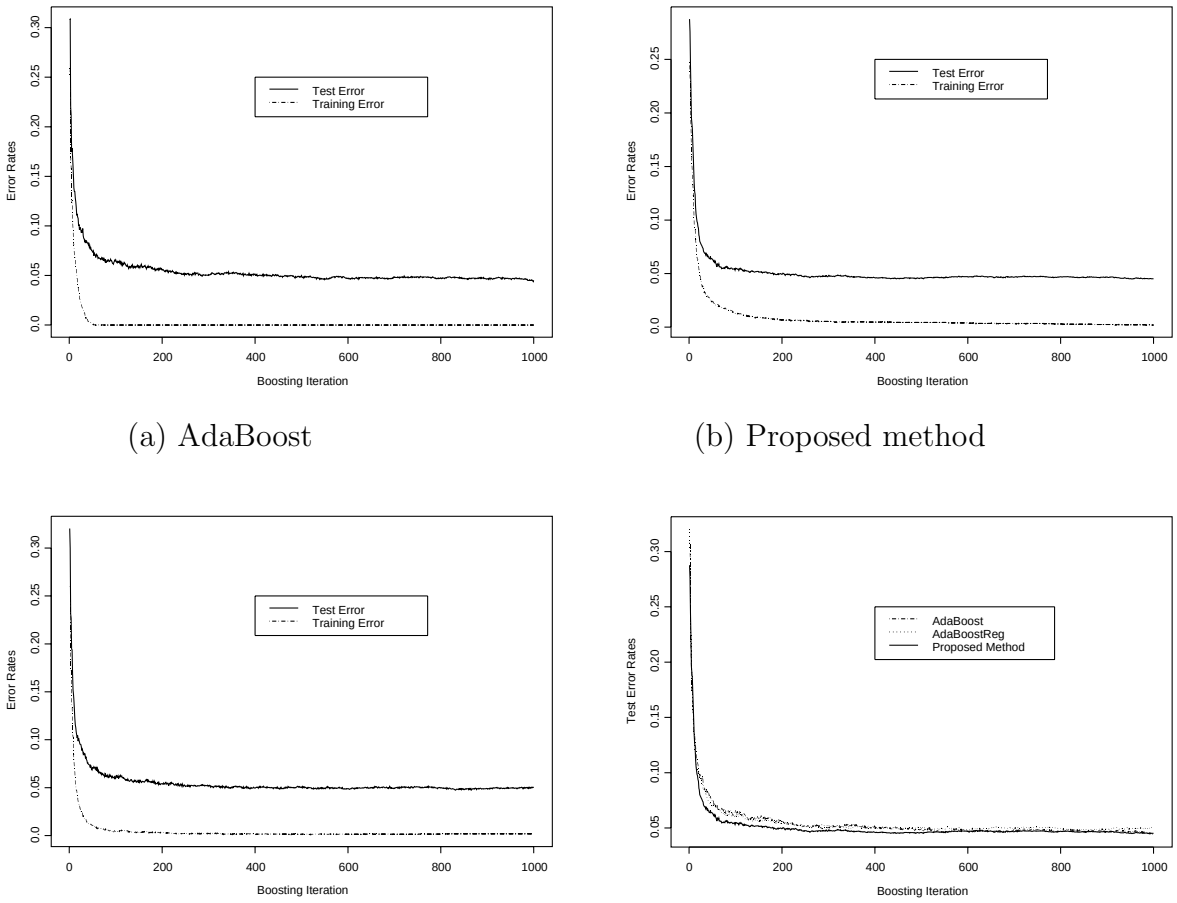


(c) AdaBoost$_{Reg}$



(d) Three methods

Figure 3: Comparison of three boosting methods on 2% mislabeled cases near decision boundary. (a) Learning and test error rates of AdaBoost. (b) Learning and test error rates of proposed method. (c) Learning and test error rates of AdaBoost$_{Reg}$. (d) Test error rates of three boosting methods.



(a) AdaBoost (b) Proposed method

(c) AdaBoost$_{Reg}$ (d) Three methods

Figure 4: Comparison of three boosting methods on noiseless cases. (a) Learning and test error rates of AdaBoost. (b) Learning and test error rates of proposed method. (c) Learning and test error rates of AdaBoost$_{Reg}$. (d) Test error rates of three boosting methods.

5. Conclusion

We developed and derived a new, robust boosting method against mislabeled, noisy data. The stagewise additive modelling methodology is blended with the gradient numerical search algorithms and is used as a design principle. The new formulation uses the smoothed zero-one sigmoidal loss function suitable for gradient descent algorithms. Performance of the proposed method was compared with that of AdaBoost and AdaBoost$_{Reg}$ through intensive numerical experiments. The proposed method is robust compared to other two boosting methods especially in mislabeled, noisy cases. Even for noiseless (0% mislabeled) cases, the proposed method exhibits the similar performance with that of other two methods. The proposed method is applicable to a wide range of problems in real-world data mining applications such as response enhancement of direct mails and improved default prediction of credit card users, etc. These are subjects for our future study.

Acknowledgments

The work of H. Suzuki and M. Koda is supported in part by a Grant-in-Aid for Scientific Research of the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

[1] L. Breiman: Combining predictors. (Technical Report, Statistics Department, University of California, Berkeley, 1998).

[2] R. O. Duda and P. E. Hart, and D. G. Stork: *Pattern Classification* (John Wiley & Sons, New York, NY, 2001).

[3] Y. Freund: Boosting a weak learning algorithm by majority. *Information and Computation*, **121** (1995), 256-285.

[4] Y. Freund and R. E. Schapire: A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, **55** (1997), 119-139.

[5] J. Friedman: Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, **29** (2001), 1189-1232.

[6] J. Friedman, T. Hastie, and R. Tibshirani: Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, **28** (2000), 337-407.

[7] A. Grove and D. Schuurmans: Boosting in the limit: maximizing the margin of learned ensembles. *Proceedings of 15th National Conference on Artificial Intelligence*, (1998), 692-699.

[8] T. Hastie, R. Tibshirani, and J. Friedman: *The Elements of Statistical Learning: Data Mining, Inference and Prediction* (Springer, 2001).

[9] M. Koda and H. Okano: A new stochastic learning algorithm for neural networks. *Journal of the Operations Research Society of Japan*, **43** (2000), 469-485.

[10] L. Mason, P. L. Bartlett, and J. Baxter: Improved generalization through explicit optimization of margins. *Machine Learning*, **38** (2000), 243-255.

[11] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling: *Numerical Recipes in C (Second Edition)* (Cambridge University Press, Cambridge, 1992).

[12] J. R. Quinlan: Boosting first-order learning. *Proceedings of 7th International Workshop on Algorithmic Learning Theory*, **1160** (1996), 143-155.

[13] G. Rätsch, T. Onoda, and K. -R. Müller: Soft margin for AdaBoost. *Machine Learning*, **42** (2001), 287-320.

[14] R. E. Schapire: The strength of weak learnability. *Machine Learning*, **5** (1990), 197-227.

Masato Koda
Graduate School of Systems and Information Engineering
University of Tsukuba
1-1-1 Tennodai
Tsukuba 305-8573, Japan
E-mail: koda@sk.tsukuba.ac.jp