INSTITUTE OF POLICY PLANNING SCIENCES
Discussion Paper Series
No. 1092

A Robust Boosting Method Using Zero-one Loss Function : SNRBoost

Natsuki Sano[†], Hideo Suzuki[††], and Masato Koda[*]
Graduate School of Systems and Information Engineering, University of Tsukuba
[†] k993208@sk.tsukuba.ac.jp [††] hsuzuki@sk.tsukuba.ac.jp [*] koda@sk.tsukuba.ac.jp

Abstract We propose a new, robust boosting method by using a zero-one step function as a loss function. In deriving the method, the MarginBoost technique is blended with the stochastic gradient approximation algorithm, called *Stochastic Noise Reaction* (SNR). Based on intensive numerical experiments, we show that the proposed method is actually better than AdaBoost on test error rates in the case of noisy, mislabeled situation.

Key words: Data mining, AdaBoost, MarginBoost.L_1, Zero-one Loss Function, Stochastic Noise Reaction.

1. Introduction–AdaBoost

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 behavior 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 [3], [4], [16]) and received increasing attention in many areas including data mining and knowledge discovery.

*Corresponding author

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 T. Then, the output of classification model (i.e., committee) is given by a scalar function, $F(x) = \sum_{t=1}^{T} \beta_t f_t(x)$, in which each $f_t(x)$ denotes a classifier producing values ± 1, and β_t are prescribed constants; the corresponding prediction (i.e., decision) is $\text{sign}(F(x))$.

The AdaBoost procedure trains the classifiers $f_t(x)$ 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 $I(y_i \neq f_t(x_i))$ denotes the indicator function with respect to the event such that $y_i \neq f_t(x_i)$, i.e., misclassification event.

At the t-th iteration stage, given the observation weight $w_{t-1}(i)$, AdaBoost fits a classifier $f_t(x)$ to the training data x_i $(i = 1, 2, \ldots, n)$ that is weighted by $w_{t-1}(i)$. Then, it evaluates the weighted error err_t by computing a weighted count of misclassification events as $\text{err}_t = \sum_{i=1}^{n} w_{t-1}(i) I(y_i \neq f_t(x_i))$. Using err_t, β_t is obtained as $\beta_t = \log((1 - \text{err}_t)/\text{err}_t)$. Finally, the observation weight is updated through the computation of $w_t(i) = w_{t-1}(i) \exp[\beta_t I(y_i \neq f_t(x_i))]$ and normalized so that $\sum_{i=1}^{n} w_t(i) = 1$. This results in giving a higher weight to the training data that is currently misclassified. A concise derivation of AdaBoost algorithm is given in Appendix A.

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 $f_t(x)$ and demonstrated that it consistently produces significantly lower error rates than a single decision tree. In fact, Breiman called AdaBoost with trees as " the best off-the-shelf classifier in the world [1]. "

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

1. Initialize the observation weights $w_1(i) = 1/n,\ i = 1, 2, \ldots, n$.
2. For $t = 1, 2, \ldots, T$ do:
   (a) Fit a classifier $f_t(x)$ to the training data using weights $w_t(i)$.
   (b) Compute $\text{err}_t = \frac{\sum_{i=1}^{n} w_t(i) I(y_i \neq f_t(x_i))}{\sum_{i=1}^{n} w_t(i)}$.
   (c) Compute $\alpha_t = \log((1 - \text{err}_t)/\text{err}_t)$.
   (d) Update weights
   $w_{t+1}(i) = w_t(i) \cdot \exp[\alpha_t \cdot I(y_i \neq f_t(x_i))],\ i = 1, 2, \ldots, n.$
   end For
3. Output $F(x) = \text{sign}[\sum_{t=1}^{T} \alpha_t f_t(x)]$.

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., [7], [13], [14], 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 [10], [14]

are proposed. 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 recapitulate the principles of MarginBoost.L_1. Then, in Section 3, we present the zero-one loss function and the gradient approximation technique referred to as Stochastic Noise Reaction (SNR). In Section 4, we propose the new robust boosting method. Results of intensive numerical experiments are presented in Section 5 where we detail cases with noisy, mislabeled patterns. Section 6 contains some concluding remarks.

2. MarginBoost.L_1

We assume that the training dataset (x, y) is randomly generated according to some unknown probability distribution $\mathcal{D}$ on $X \times Y$ where X is the space of measurements (typically $X \subseteq \mathbb{R}^n$) and Y is the space of labels (Y is usually a discrete set $\{\pm 1\}$ or some subset of $\mathbb{R}$). The output of classification is denoted as $\text{sign}(F(x))$, where $F(x)$ is given by

$$F(x) = \sum_{t=1}^{T} \beta_t f_t(x), \tag{1}$$

and $f_t(x) : X \to \{\pm 1\}$ are base classifiers from some fixed class $\mathcal{F}$ (e.g., class of base hypotheses), and $\beta_t \in \mathbb{R}$ denotes the weights.

The *margin* of the training data (x, y) with respect to the classifier is defined as $z = yF(x)$. It is noted that if $y \neq \text{sign}(F(x))$ then $yF(x) < 0$, and if $y = \text{sign}(F(x))$ then $yF(x) \geq 0$. The margin can be understood as a measure of difficulty of classification. As data has a larger (positive) margin, the example is easier to classify. On the contrary, as it has a smaller (negative) margin, the data is harder to classify.

Given a set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of n labeled data pairs generated according to $\mathcal{D}$, we wish to construct a classification model described by Equation (1) so that the probability of incorrect classification $P_{\mathcal{D}}(\text{sign}(F(x)) \neq y)$ is small. Since $\mathcal{D}$ is unknown and we are only given a training set S, we take the approach of finding the classification model which minimizes the sample risk of some cost function of the margin. That is, for a training set S we want to find F such that the empirical average,

$$L(F) = \frac{1}{n} \sum_{i=1}^{n} C(y_i F(x_i)) \tag{2}$$

is minimized for some suitable cost function $C(z) : \mathbb{R} \to \mathbb{R}$. The interpretation of AdaBoost as an algorithm which performs a gradient descent optimization of the sample risk has been examined by several authors, e.g., [5], [11], [15].

Mason et al. [11] proposed MarginBoost which gives a general framework of AdaBoost in terms of the margin. AdaBoost can be seen as a special case of MarginBoost with $C(z_i^t) = \exp(-z_i^t)$, where z_i^t denotes the margin, $z_i^t = y_i F_t(x_i)$. In the next box, the normalized version of MarginBoost, MarginBoost.$L1$ is summarized, in which β_s are normalized such that they sum to one.

In the box, $C'(z)$ denotes the derivative of the cost function with respect to the margin z. Note that $C'(z)$ is non-positive since the margin cost function is monotonically decreasing and the normalized weight $w_t(i) = \frac{C'(z_i^t)}{\sum_{j=1}^{n} C'(z_j^t)} > 0$ can be interpreted as a probability, but it actually gives a gradient information, which will serve as a basis for deriving a gradient-type

MarginBoost.L_1 (Mason et al., [11])

1. Specify a suitable (monotonically decreasing) cost function $C(z)$.
2. Initialize $w_0(i) = 1/n$ for $i = 1, 2, \ldots, n$, and $F_0(x) = 0$.
3. For $t = 1, 2, \ldots, T$, do:
   (a) Fit a classifier $f_t(x)$ to the training data weighted by $w_{t-1}(i)$ for $i = 1, 2, \ldots, n$.
   (b) If $\sum_{i=1}^{n} w_{t-1}(i) y_i f_t(x_i) \geq 0$, then return $F_t(x)$, and stop. end If.
   (c) Choose β_t appropriately.
   (d) Let $F_t(x) = F_{t-1}(x) + \beta_t f_t(x)$.
   (e) Set $w_t(i) = \frac{C'(z_i^t)}{\sum_{j=1}^{n} C'(z_j^t)}$ for $i = 1, 2, \ldots, n$.
   end For
4. Output $\text{sign}(F_T(x))$.

search algorithm in the subsequent development. If $\sum_{i=1}^{n} w_{t-1}(i) y_i f_t(x_i) \geq 0$, i.e., when the total sum of the weighted margins indicates correct classification over the training data, then the algorithm returns $F_t(x)$ and terminates. The readers are referred to [11] for details of MarginBoost.L_1.

3. Misclassification Loss Function and Stochastic Noise Reaction

3.1. Misclassification (Zero-One) Loss Function

Friedman et al. [6] have demonstrated that the AdaBoost algorithm decreases an exponential loss function. Figure 1 illustrates various loss functions as a function of the margin value, $z = yF(x)$, 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 dimension.

Shown also in Figure 1 is the misclassification loss (zero-one loss function), $I(z < 0) = I(yF(x) < 0)$, where $I(yF(x) < 0)$ denotes the indicator function with respect to the occurrence of the incorrect events, i.e., $yF(x) < 0$, which gives unit penalty for negative margin values, with no penalty for positive ones (i.e., correct decisions). In this way, the decision rule becomes a judgment on a zero-one loss function, which will be investigated in this study. It is important to note that the zero-one loss function yields the Bayes minimum classification error for binary classification [2]. In order to approximate the misclassification loss, Sano et al. [15] proposed the sigmoidal loss function (i.e., smoothed zero-one loss function, $C(z) = 1/(1 + \exp(\lambda z))$, where λ denotes the positive slope parameter), since the misclassification loss is a discontinuous step function. In this study, however, the derivative of the misclassification loss is stochastically approximated without using the sigmoidal 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 misspeci-

fication of the class labels in the training data. Since the zero-one loss function 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.

Mean-squared approximation errors are well-understood and used as a loss function in statistics area. Unlike the zero-one loss function which considers only the misclassified 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 $yF(x) > 1$ incurs increasing loss for larger values of $|F(x)|$. This makes the squared-error a poor approximation compared to the zero-one loss function 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.

Here we would like to propose the zero-one loss function which takes account of limited influences from larger negative margins in a proper manner and, accordingly, robust against mislabeled noisy training data. Actual misclassification loss is not differentiable at $yF(x) = 0$, and one needs to use sub-gradients in general. However, in this study, we try to avoid formulating sub-gradients, and we use *Stochastic Noise Reaction* (SNR) technique proposed in [9] in order to evaluate the derivative of the zero-one loss function.

3.2. Stochastic Noise Reaction

Optimization problems that seek for minimization (or equivalently maximization) of an objective function have practical importance in various areas. Once a task is modeled as an optimization problem, general optimization techniques become applicable; e.g., linear programming, gradient methods, etc. One may encounter difficulties, however, in applying these techniques when the objective function is non-differentiable or when it is defined as a procedure. To deal with such a case, Koda and Okano [9] proposed a function minimization algorithm, called Stochastic Noise Reaction (SNR) that updates solutions based on approximated derivative information. They apply SNR algorithm to *traveling salesman problem* (TSP), a representative combinatorial optimization problem, and show its effectiveness [12]. We illustrate here only an essence of SNR needed in the present boosting study.

Consider an objective function $L(z) : \mathbb{R}^n \to \mathbb{R}$, then the gradient vector of $L(z)$ is defined by

$$\nabla L(z) = \left(\frac{\partial L(z)}{\partial z_1}, \frac{\partial L(z)}{\partial z_2}, \cdots, \frac{\partial L(z)}{\partial z_n} \right)^T. \tag{3}$$

To avoid the exact gradient calculation of Equation (3), SNR technique injects a Gaussian white noise, $\xi_i \in N(0, \sigma_i^2)$, into a variable z_i as

$$z_i(j) = z_i + \xi_i(j), \tag{4}$$

where $\xi_i(j)$ denotes the j-th realization of the noise injected into the i-th variable. Each component of a derivative is approximated for sufficiently small $\sigma_i < \sqrt{2}$, without explicitly differentiating the objective function by using the relation,

$$\frac{\partial L(z)}{\partial z_i} \approx \frac{1}{\sigma_i^2} \frac{1}{M} \sum_{j=1}^{M} L(z(j)) \xi_i(j) = d(z_i), \tag{5}$$

where M is a loop count for taking the average. For details of the derivation of Equation (5), the readers are referred to [12]. A concise derivation of Equation (5) is given in Appendix

B. It is expected that Equation (5) yields a good approximation when $\sigma_i \to 0$ (cf. Equation (B3) in Appendix B). The value of M is set to 100 in all of the numerical experiments in Section 5. In actual implementations, all the realizations of the noise should be generated and normalized in advance to ensure sample mean $\frac{1}{M}\sum_{j=1}^{M}\xi_i(j) = 0$ and sample variance $\frac{1}{M}\left(\xi_i(j) - \frac{1}{M}\sum_{j=1}^{M}\xi_i(j)\right)^2 = \sigma_i^2$.

3.3. SNR Application to Zero-one Loss Function

Based on Equation (5) with $\sigma_i = 1$, the approximated derivative for the misclassification loss, i.e., zero-one loss function,

$$I(z < 0) = \begin{cases} 1 & \text{if } z < 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

is shown in Figure 2. Figure 2 depicts that the discontinuous point of the zero-one loss function at $z = 0$ is leveled, and the derivative is obtained. Note that the stochastic gradient resembles the derivative of $C(z) = \frac{1}{2}(1 - \tanh(z))$, which is the smoothed zero-one loss function and is differentiable as $C'(z) = \frac{1}{2}(\tanh(z) + 1)(\tanh(z) - 1)$. It is recognized that the stochastic gradient is negative and smallest at the origin, and equals to 0 as z is distant from the origin. It is difficult to observe, however, that there could be a positive derivative at distant points from the origin.

4. Proposed Boosting Method–SNRBoost

Based on the algorithmic framework of MarginBoost.L_1 and Stochastic Noise Reaction (SNR), the proposed boosting method referred to as SNRBoost is presented in this section. Essentially the proposed algorithm works as in MarginBoost.L_1, however, in deriving the method, we use the zero-one loss function (cf. Equation (6)) as the margin cost function in finding pattern weightings for the training set.

In principle, any boosting algorithm selects iteratively a base learner or hypothesis at a time and then updates the weight w. In MarginBoost.L_1, one updates only the weight of the last classifier selected as described in the line 3(e) of MarginBoost.L_1 algorithm as follows:

$$w_t(i) = \frac{C'(z_i^t)}{\sum_{j=1}^{n} C'(z_j^t)}, \tag{7}$$

where z_i^t denotes the margin of the i-th training data at the t-th iteration, $z_i^t = y_i F_t(x_i)$, and $C(z_i^t)$ is the differentiable (monotonically decreasing) cost function. Recall that $C'(z_i^t) < 0$ and the normalized weight $w_t(i) > 0$ can be interpreted as a probability, but it actually gives a gradient information.

Let $d_t(i)$ denote the stochastic gradient approximated by using SNR algorithm (cf. Equation (5)) for the zero-one loss function at z_i^t. Then, we may incorporate SNR algorithm in updating the weight by replacing the derivatives $C'(z_j^t)$ involved in Equation (7), utilizing a formal relationship $C'(z_j^t) = d_t(j)$ even when the margin cost function $C(z)$ is non-differentiable, zero-one step function, i.e., $C(z) = I(z < 0)$.

Therefore, in view of our ultimate goal to achieve a gradient-type minimization through the zero-one loss function, we propose to update the pattern weighting as follows:

$$w_t(i) = \frac{d_t(i)}{\sum_{j=1}^{n} d_t(j)}. \tag{8}$$

Note, however, Equation (8) may result in computing negative weights due to a stochastic nature of SNR algorithm and $w_t(i) \geq 0$ is not automatically satisfied. Hence, if $w_t(i) < 0$, then it is set to 0 and the computing process is continued to obtain weightings of the training set for a selection of the next classifier which minimizes the weighted training error. If the total sum of the weighted margins becomes non-negative meaning correct classification over the training data (i.e., $\sum_{i=1}^{n} w_{t-1}(i) y_i f_t(x_i) \geq 0$), then the algorithm returns $F_t(x)$ and terminates.

Above weighting rule implies that the weight is assigned proportional to the normalized value of the corresponding gradient computed from Equations (7) or (8). For instance, the largest weight is given to the training data with largest gradient component (related to the quantity often called the *edge* in supervised learning [1]); this is easily understood as the training data with greatest importance is the one that makes most progress in the gradient-type search. Hence, MarginBoost.L_1 and SNRBoost both fall into the category of gradient-type boosting algorithms.

For the appropriate choice of β_t in Equation (1), which also appears in the lines 3(c) and 3(d) of MarginBoost.L_1 algorithm, we propose $\beta_t = K/(K+t)$, where K is a suitable integer. This setting of β_t is based on the well-known stochastic approximation technique [2], which satisfies the conditions

$$\lim_{T\to\infty} \sum_{t=1}^{T} \beta_t = \infty, \text{ and } \lim_{T\to\infty} \sum_{t=1}^{T} \beta_t^2 < \infty. \tag{9}$$

Note, if the technique is used in the gradient-type search, the proposed method guarantees a convergence of F_t to its local optimal point. In our numerical experiments in Section 5, the proposed method of setting β_t with $K = T$ is used in the implementation of MarginBoost.L_1 and SNRBoost.

We have now discussed all the necessary parts of the proposed algorithm, which is summarized in the following box as pseudo-code.

SNRBoost

1. Initialize $w_0(i) = 1/n$ for $i = 1, 2, \ldots, n$, $F_0(x) = 0$ and specify K.
2. For $t = 1, 2, \ldots, T$, do:
   (a) Fit a classifier $f_t(x)$ to the training data weighted by $w_{t-1}(i)$ for $i = 1, 2, \ldots, n$.
   (b) If $\sum_{i=1}^{n} w_{t-1}(i) y_i f_t(x_i) \geq 0$, then return $F_t(x)$, and stop.
   end If.
   (c) Set $\beta_t = K/(K+t)$.
   (d) Let $F_t(x) = F_{t-1}(x) + \beta_t f_t(x)$.
   (e) Compute approximated derivatives for zero-one loss function $d_t(i)$ using SNR for $i = 1, 2, \ldots, n$.
   (f) Set $w_t(i) = \frac{d_t(i)}{\sum_{j=1}^{n} d_t(j)}$ for $i = 1, 2, \ldots, n$.
   (g) If $w_t(i) < 0$ then $w_t(i) = 0$.
   end If.
   end For
3. Output sign$(F_T(x))$.

5. Numerical Experiments

In this section, numerical results are presented and, especially, the robustness of the proposed method are analyzed and compared with that of AdaBoost and MarginBoost.L_1 with $C(z) = \frac{1}{2}(1 - \tanh(\lambda z))$, where λ is the positive slope parameter. The normalized version of AdaBoost and MarginBoost.L_1 such that $\sum_{s=1}^{t} \beta_s = 1$ is used to compare effects of the loss function. We focus our attention to classification results for mislabeled (i.e., noisy) cases. The back-propagation neural network with single hidden layer is used as a base learner for all the three boosting methods. The number of units in hidden layer (i.e., hidden units) is 3. Since 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, it thus provides the ideal potential for generalization of training results by boosting. Throughout the experiments, SNR algorithm uses an estimator of the gradients which averages 100 noisy samples, i.e., $M = 100$ (cf. Equation (5)).

5.1. Toy Example

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

1. Generate uniformly $\mathbf{x} = (x_1, x_2) \in \chi = [-4, 4] \times [-4, 4]$;
2. assign $y = \mathrm{sign}(F(\mathbf{x}))$, where $F(\mathbf{x}) = x_2 - 3\sin(x_1)$;
3. sort $|F(\mathbf{x}_i)|$ by descending order;
4. sample randomly 2% from top 50% (far case) or bottom 50% (near case) examples;
5. flip sampled examples in Step 4.

In Step 2, note that $F(\mathbf{x}) = x_2 - 3\sin(x_1)$ is used as a nonlinear decision function. In Step 4, we generate two mislabeled cases; one where mislabeled data are located far from the decision boundary (far case) and the other where mislabeled data are concentrated near the decision boundary (near case). Figure 3(a) shows the far case, while Figure 3(b) shows the near case, and Figure 3(c) shows the noiseless case. The decision boundary is given by the solid line.

The number of training data is 300 and that of test data is 1000. In all the experiments, noiseless data is used as test data. Iteration number of the base learner, which is referred to as round (number), is set to 1000, and the stopping condition is not used to compare effects of the margin cost function utilized. Above procedure is repeated 5 times, each time different dataset is generated and the performance of boosting methods for each dataset is evaluated. Figure 4 shows the average performance of test error rates. Variance of the proposed SNR method is set to $\mathrm{var}(\xi_i) = 1.0$, and for MarginBoost.$L_1$, λ is set to $\lambda = 1.0$. In all three cases, the performance of SNRBoost exhibits similar performance to MarginBoost.L_1 with $C(z) = \frac{1}{2}(1 - \tanh(z))$. It is clear that the performance of SNRBoost is superior to AdaBoost in mislabeled cases (see Figures 4(a) and 4(b)).

5.2. Higher-dimensional Example

In this experiment, the level of variance of the proposed SNR method is changed as $\mathrm{var}(\xi_i) = 1.0,\ 0.1,\ 0.01$, and MarginBoost.$L_1$ is executed using the range of slope parameters with $\lambda = 1,\ 5,\ 10$. 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 the correctly labeled (i.e., noiseless) case, as in toy data experiments.

5.2.1. 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 n training and n 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 bottom) 50% of 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 far from decision boundary, samples from top 50% training example are selected in Step 7. In the mislabeled case near decision boundary, samples from bottom 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 dataset is generated and the performance of boosting methods for each dataset is evaluated. The number of training and test data is 1000 each. Iteration number of the base learner is 1000, and the stopping condition is not used to compare effects of the margin cost function.

5.2.2. Numerical Results

We plot in Figure 5 the test error curves for 2% mislabeled data located far from the decision boundary, and in Figure 6 that for 2% mislabeled data near the decision boundary. In Figure 7, the test error curves are shown for noiseless case. In each figure, (a) shows AdaBoost test error rates, (b) test error rates of the proposed method, with $\mathrm{var}(\xi_i) = 1.0,\ 0.1,\ 0.01$, (c) test error rates of the MarginBoost.L_1 with $\lambda = 1,\ 5,\ 10$, and (d) comparison of test error rates of AdaBoost, SNRBoost and MarginBoost.L_1, respectively.

(i) Mislabeled case far from decision boundary

In Figure 5(a), AdaBoost test error rate falls into 0.105% at 1000 round (boosting iteration). In Figure 5(b), for SNRBoost with $\mathrm{var}(\xi_i) = 0.01$, test error rate falls into 0.048% at 1000 round. In Figure 5(c), for MarginBoost.L_1 with $\lambda = 10$, test error rate falls into 0.048% at 1000 round. In Figure 5(d), on test error rates, the performance of SNRBoost with $\mathrm{var}(\xi_i) = 0.01$ is superior to that of AdaBoost and exhibits similar performance to MarginBoost.L_1 with $\lambda = 10$.

(ii) Mislabeled case near decision boundary

In Figure 6(a), AdaBoost test error rate falls into 0.070% at 1000 round. In Figure 6(b), for SNRBoost with $\mathrm{var}(\xi_i) = 0.01$, test error rate falls into 0.049% at 1000 round. In Figure 6(c), for MarginBoost.L_1 with $\lambda = 10$, test error rate falls into 0.053% at 1000 round. In Figure 6(d), on test error rates, the performance of SNRBoost with $\mathrm{var}(\xi_i) = 0.01$ is superior to that of AdaBoost and exhibits slightly better performance than that of MarginBoost.L_1 with $\lambda = 10$.

(iii) Noise-free case

In Figure 7(a), AdaBoost test error rate falls into 0.048% at 1000 round. In Figure 7(b), for SNRBoost with $\mathrm{var}(\xi_i) = 0.01$, test error rate falls into 0.038% at 1000 round. In Figure

7(c), for MarginBoost.L_1 with $\lambda = 10$, test error rate falls into 0.044% at 1000 round. In Figure 7(d), the performance of SNRBoost with $\mathrm{var}(\xi_i) = 0.01$ is superior to that of AdaBoost and exhibits better performance than that of Margin.Boost.L_1 with $\lambda = 10$.

From the above, it is found that the performance of the proposed SNRBoost is improved when $\mathrm{var}(\xi_i)$ is smaller, while that of MarginBoost.L_1 with $C(z) = \frac{1}{2}(1 - \tanh(\lambda z))$ is improved as λ becomes larger. We note that the proposed margin cost function for MarginBoost.L_1 approaches the zero-one step function as λ becomes larger, which may validate the use of the zero-one loss function in the present method. As expected, SNR method gives improved accuracy when $\mathrm{var}(\xi_i)$ is smaller (cf. Equation (B3) in Appendix B). In summary, SNRBoost exhibits a higher potential for generalization than AdaBoost.

6. Conclusion

We developed a new, robust boosting method against mislabeled, noisy data. The new formulation uses the misclassification loss function, i.e., zero-one step function. In deriving the algorithm, Stochastic Noise Reaction technique [9] is used to approximate the gradient of the zero-one loss function. Performance evolution (i.e., error minimization) of the proposed method is compared with that of AdaBoost and MarginBoost.L_1 with $C(z) = \frac{1}{2}(1 - \tanh(\lambda z))$ through intensive numerical experiments. The proposed method is robust compared to AdaBoost especially in the mislabeled cases. Even for the mislabeled data located far from decision boundary, the method exhibits the similar performance to that of MarginBoost.L_1.

Acknowledgements

We thank Hiroyuki Okano at IBM Tokyo Research Laboratory, for useful comments on an earlier draft. The work of M. Koda and H. Suzuki is supported in part by a Grant-in-Aid for Scientific Research of the Ministry of Education, Culture, Sports, Science and Technology of Japan.

Appendices

A. Derivation of AdaBoost

Consider the following exponential loss function:

$$C(y, F(x)) = \exp(-yF(x)), \tag{A1}$$

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 (A1), one must solve

$$(\beta_t, f_t) = \arg\min_{\beta, f} \sum_{i=1}^{n} \exp[-y_i(F_{t-1}(x_i) + \beta f(x_i))]$$

for the optimal classifier $f_t(x)$ and corresponding coefficient β_t to be added at the t-th step. This can be expressed as

$$(\beta_t, f_t) = \arg\min_{\beta, f} \sum_{i=1}^{n} w_t(i) \exp(-\beta y_i f(x_i)) \tag{A2}$$

with $w_t(i) = \exp(-y_i F_{t-1}(x_i))$. Since each $w_t(i)$ depends neither on β nor $f(x)$, it can be regarded as a weight that is applied to each observation. This weight depends on $F_{t-1}(x_i)$, and so the individual weight value changes at each iteration t.

The solution to Equation (A2) can be obtained in two steps. First, for any value of $\beta > 0$, the solution to Equation (A2) for $f_t(x)$ is given by

$$f_t(x) = \arg\min_{f} \sum_{i=1}^{n} w_t(i) I(y_i \neq f(x_i)), \tag{A3}$$

which is the classifier that minimizes the weighted error rate in predicting y. This can be easily seen by expressing the criterion in Equation (A2) as

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

which in turn can be written as

$$(e^{\beta} - e^{-\beta}) \cdot \sum_{i=1}^{n} w_t(i) I(y_i \neq f(x_i)) + e^{-\beta} \cdot \sum_{i=1}^{n} w_t(i). \tag{A4}$$

By plugging in $f = f_t$, i.e., Equation (A3), into Equation (A2) and solving for the optimal value of β, one obtains

$$\beta_t = \frac{1}{2} \log \frac{1 - \mathrm{err}_t}{\mathrm{err}_t}, \tag{A5}$$

where err_m is the optimal error rate defined as

$$\mathrm{err}_t = \frac{\sum_{i=1}^{n} w_t(i) I(y_i \neq f_t(x_i))}{\sum_{i=1}^{n} w_t(i)}. \tag{A6}$$

Note that Equation (A6) is equivalent to line 2(b) of AdaBoost.M1. algorithm. The representation of $F_t(x)$ is then updated as

$$F_t(x) = F_{t-1}(x) + \beta_t f_t(x),$$

which renews the weights for the next iteration as follows:

$$w_{t+1}(i) = w_t(i) \cdot e^{-\beta_t y_i f_t(x_i)}. \tag{A7}$$

Using the fact that $-y_i f_t(x_i) = 2 \cdot I(y_i \neq f_t(x_i)) - 1$, Equation (A7) becomes

$$w_{t+1}(i) = w_t(i) \cdot e^{\alpha_t I(y_i \neq f_t(x_i))} \cdot e^{-\beta_t}, \tag{A8}$$

where $\alpha_t = 2\beta_t$ is the quantity defined at line $2(c)$ of AdaBoost.M1. The factor $e^{-\beta_t}$ in Equation (A8) multiplies all weights by the same value, so it has no effect. Thus Equation (A8) 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 weighted minimization involved in Equation (A3). Hence we conclude that AdaBoost.M1. minimizes the exponential loss criterion (A1) through adaptively adding a new function $f_t(x)$.

B. Derivation of Equation (5)

The Stochastic Noise Reaction (SNR) algorithm, Equation (5), uses the following relationship:

$$\frac{\partial L(z)}{\partial z_i} \approx \frac{1}{\sigma_i^2} \langle L(z+\xi)\xi_i \rangle, \tag{B1}$$

where $L(z) : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function, and $\langle \cdot \rangle$ denotes the expectation operator. In Equation (B1), ξ is an n-dimensional vector each of whose components ξ_i is an independent Gaussian noise with mean 0 and standard deviation σ_i; i.e.,

$$\langle \xi_i \xi_j \rangle = \sigma_i^2 \delta_{ij}, \quad \langle \xi_i^{2r+1} \rangle = 0, \quad \langle \xi_i^{2r} \rangle = \sigma_i^{2r} \frac{(2r)!}{2^r r!}, \tag{B2}$$

where δ_{ij} denotes the Kronecker delta, and r is a non-negative integer value.

Using (B2) in a Taylor series expansion of $L(z+\xi)$ around z gives

$$\begin{aligned}
\tfrac{1}{\sigma_i^2}\langle L(z+\xi)\xi_i \rangle &= \tfrac{1}{\sigma_i^2}\langle \{L(z) + \textstyle\sum_{k=1}^{\infty} \frac{1}{k!} \sum_{j=1}^{n} (\xi_j \frac{\partial}{\partial z_j})^k L(z)\}\xi_i \rangle \\
&= \tfrac{1}{\sigma_i^2}\{L(z)\langle \xi_i \rangle + \textstyle\sum_{k=1}^{\infty} \frac{1}{k!} \sum_{j=1}^{n} \langle \xi_i (\xi_j \frac{\partial}{\partial z_j})^k \rangle L(z)\} \\
&= \tfrac{1}{\sigma_i^2} \textstyle\sum_{k=1}^{\infty} \frac{1}{k!} \langle \xi_i^{k+1} \rangle \frac{\partial^k}{\partial z_i^k} L(z) \\
&= \tfrac{1}{\sigma_i^2} \textstyle\sum_{k=2,4,6,\ldots} \frac{1}{(k-1)!} \langle \xi_i^k \rangle \frac{\partial^{k-1}}{\partial z_i^{k-1}} L(z) \\
&= \tfrac{\langle \xi_i^2 \rangle}{\sigma_i^2} \tfrac{\partial L(z)}{\partial z_i} + \textstyle\sum_{k=4,6,8,\ldots} \frac{1}{(k-1)!} \frac{\langle \xi_i^k \rangle}{\sigma_i^2} \frac{\partial^{k-1}}{\partial z_i^{k-1}} L(z) \\
&= \tfrac{\partial L(z)}{\partial z_i} + \textstyle\sum_{r=2}^{\infty} \frac{1}{(r-1)!} \left(\frac{\sigma_i}{\sqrt{2}} \right)^{2(r-1)} \frac{\partial^{2r-1}}{\partial z_i^{2r-1}} L(z) \\
&\approx \tfrac{\partial L(z)}{\partial z_i},
\end{aligned} \tag{B3}$$

for sufficiently small $\sigma_i < \sqrt{2}$. It is expected that Equation (B3) yields a good approximation when $\sigma_i \to 0$. In Equation (B3), it is important to note that the differential operator has disappeared and the gradient information is given as an expected value of the product of the cost function $L(z+\xi)$ and the noise ξ_i. Since the present method will smoothen the margin cost function to be estimated by sampling and averaging, it may be efficiently applied to discontinuous functions, such as piecewise constant functions or zero-one loss function. In Equation (5), the expectation is replaced by the sample mean.

References

[1] L. Breiman, Combining Predictors, Technical Report, Statistics Department, University of California, Berkeley, 1998.

[2] R.O. Duda, 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 (2)(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 (1)(1997)119-139.

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

[6] J. Friedman, T. Hastie., and R. Tibshirani, Additive logistic regression: a statistical view of boosting, *Annals of Statistics* 28 (2)(2000)337-407.

[7] A. Grove and D. Schuurmans, Boosting in the limit: Maximizing the margin of learned ensembles, Proc. 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, New York, NY, 2001.

[9] M. Koda and H. Okano, A new stochastic learning algorithm for neural networks, Journal of the Operations Research Society of Japan 43 (4)(2000)469-485.

[10] L. Mason, P. L. Bartlett, and J. Baxter, Improved generalization through explicit optimization of margins, Machine Learning 38 (3)(2000)243-255.

[11] L. Mason, J. Baxter, P.L. Bartlett, and M. Frean, Functional gradient techniques for combining hypotheses. In A.J.Smola, P.L.Bartlet, B.Scholköpt, and D. Schuurmans, editors, Advances in Large Margin Classifiers, MIT Press, Cambridge, MA, 2000, pp. 221-246.

[12] H. Okano and M. Koda, An optimization algorithm based on stochastic sensitivity analysis for noisy objective landscapes, Reliability Engineering and System Safety 79 (2)(2003)245-252.

[13] J.R. Quinlan, Boosting first-order learning, Proc. 7th International Workshop on Algorithmic Learning Theory 1160 (1996)143-155.

[14] G. Rätsch, T. Onoda, and K.-R. Müller, Soft margin for AdaBoost, Machine Learning 42 (3)(2001)287-320.

[15] N. Sano, H. Suzuki and M. Koda, A robust boosting method for mislabeled data, Journal of the Operations Research Society of Japan 47 (3)(2004)182-196.

[16] R.E. Schapire, The strength of weak learnability, Machine Learning 5 (2)(1990)197-227.

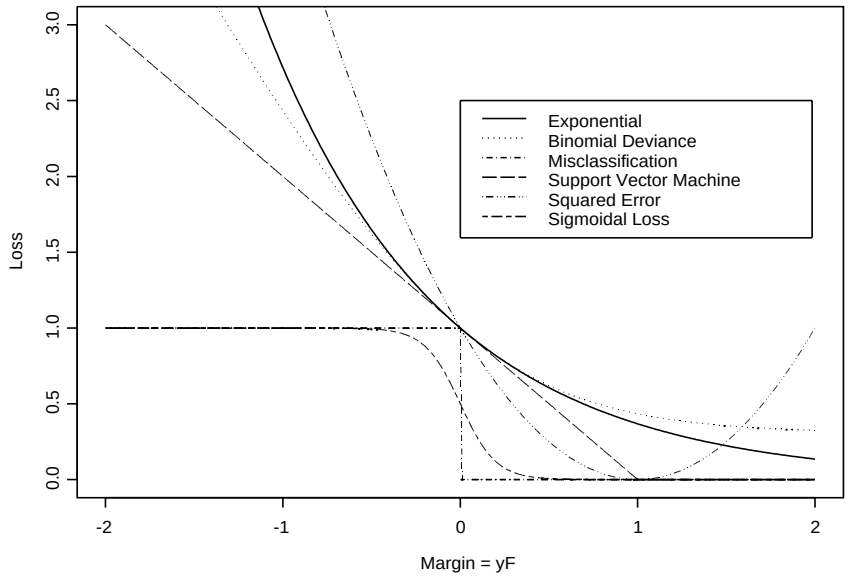


Figure 1: Loss functions for two-category classification (Following [8])

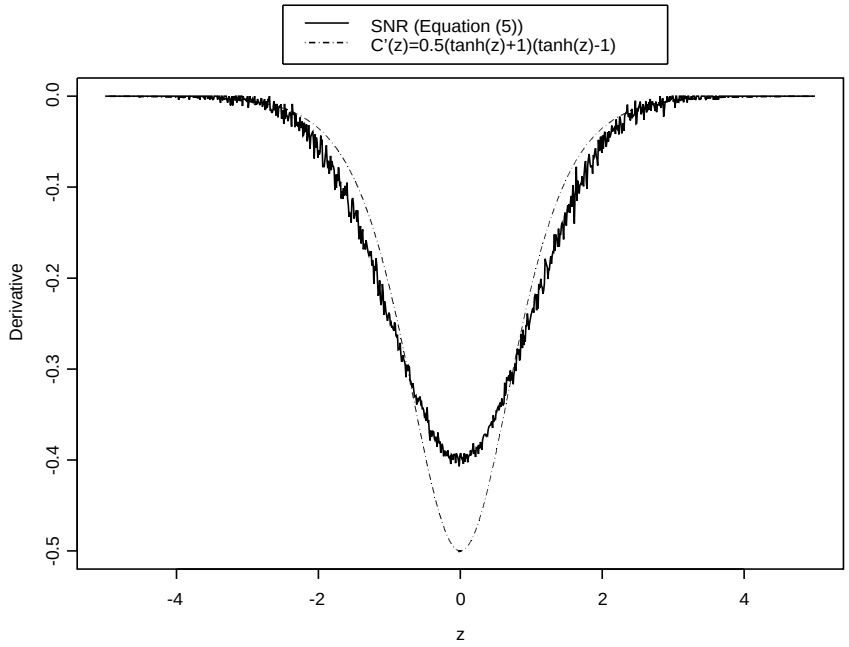
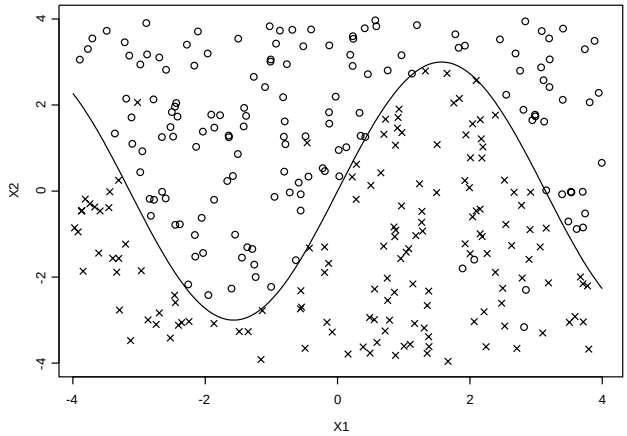
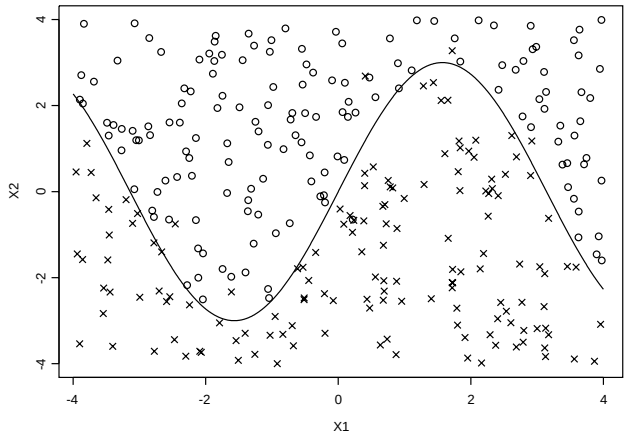


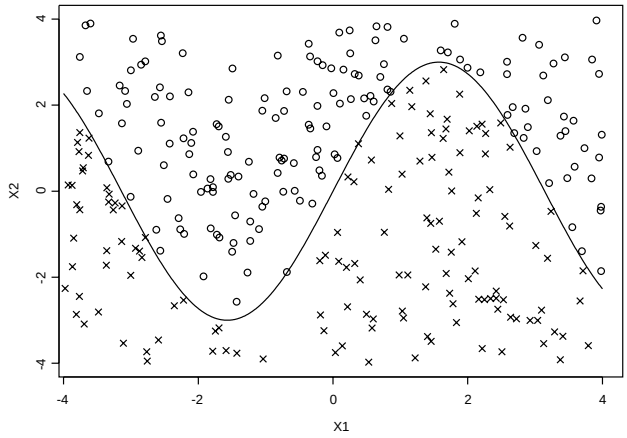
Figure 2: Derivative approximation for zero-one loss function



(a) Far Case

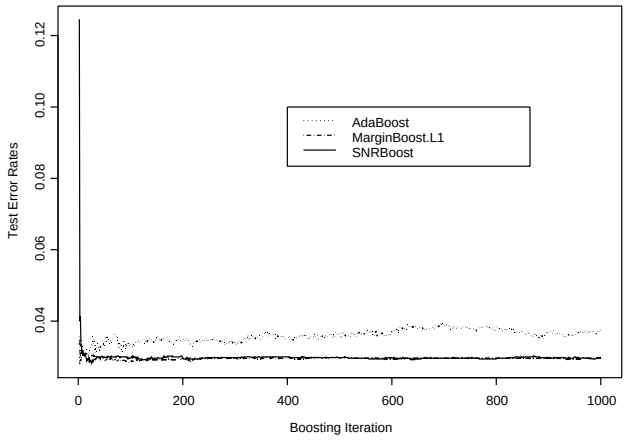


(b) Near Case

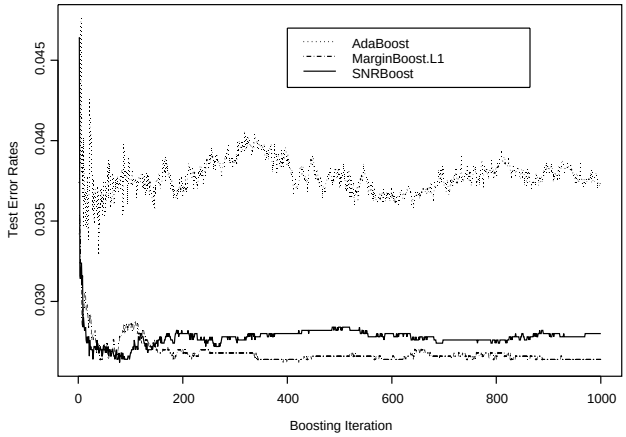


(c) Noiseless Case

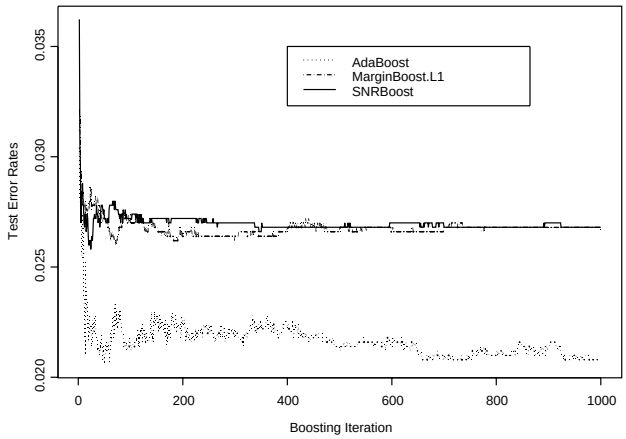
Figure 3: Toy Example
(a) Case of 2% mislabeled data located far from the decision boundary. (b) Case of 2% mislabeled data concentrated near the decision boundary. (c) Noiseless data.



(a) Far Case

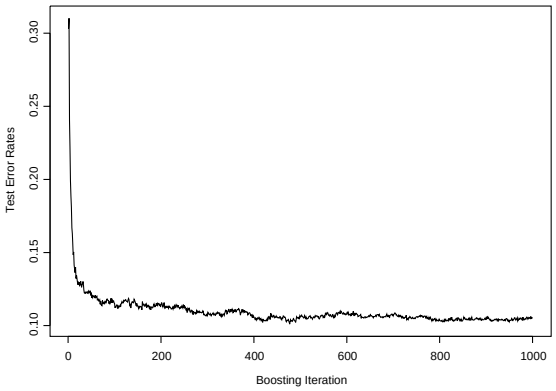


(b) Near Case

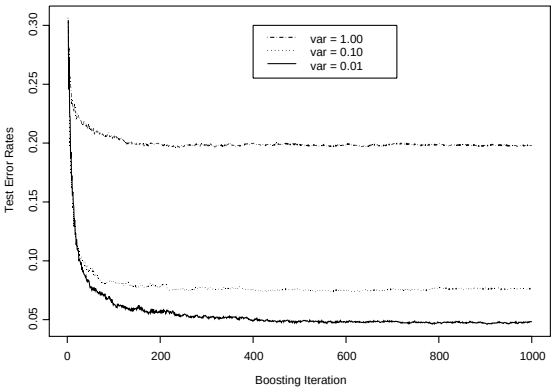


(c) Noiseless Case

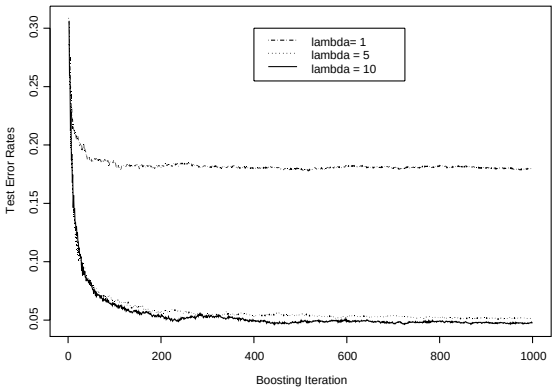
Figure 4: Comparison of three boosting methods for toy example. (a) Test error rates of three boosting methods in 2% mislabeled data located far from the decision boundary. (b) Test error rates of three boosting methods in 2% mislabeled data concentrated near the decision boundary. (c) Test error rates of three boosting methods in noiseless data.



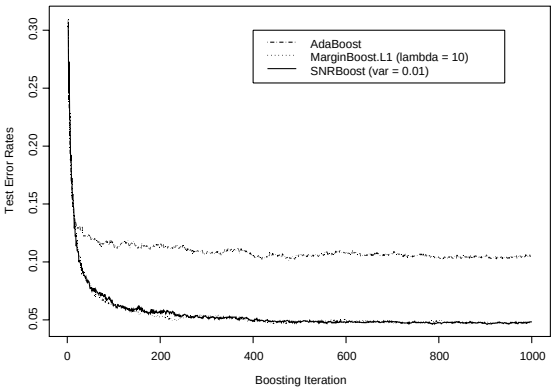
(a) AdaBoost



(b) SNRBoost

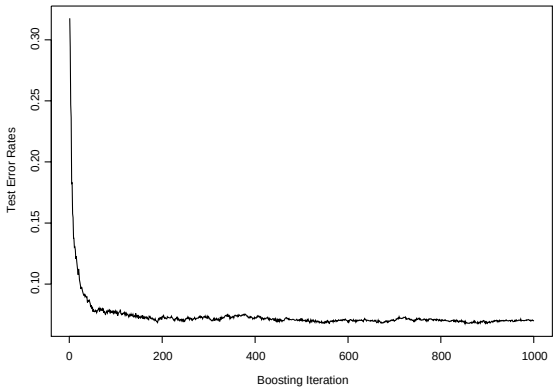


(c) MarginBoost.L_1

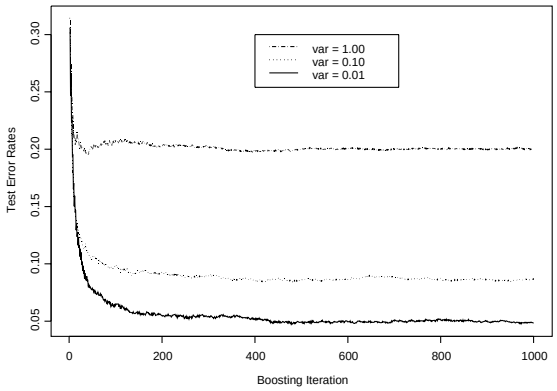


(d) Three methods

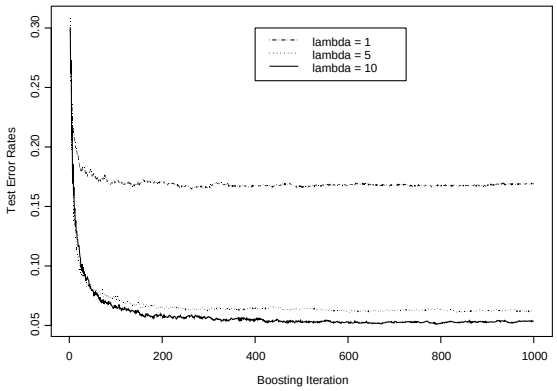
Figure 5: Comparison of three boosting methods for 2% mislabeled data far from decision boundary. (a) Test error rates of AdaBoost. (b) Test error rates of SNRBoost. (c) Test error rates of MarginBoost.L_1. (d) Test error rates of three boosting methods.



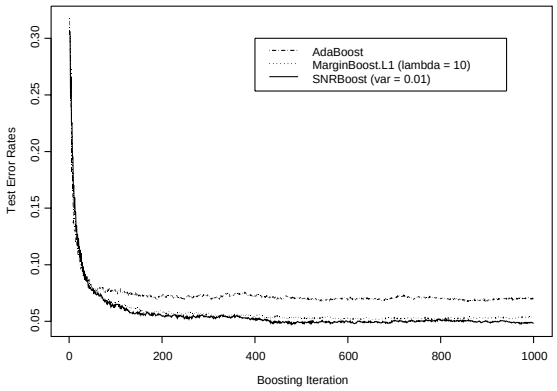
(a) AdaBoost



(b) SNRBoost

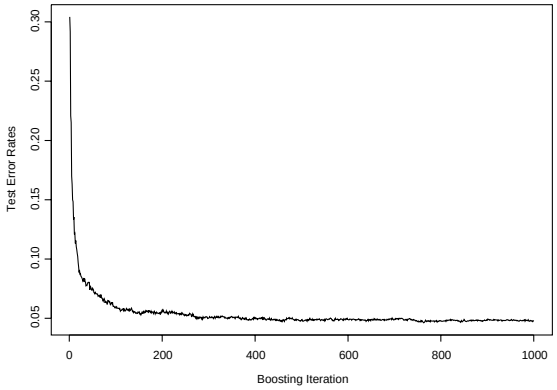


(c) MarginBoost.L_1

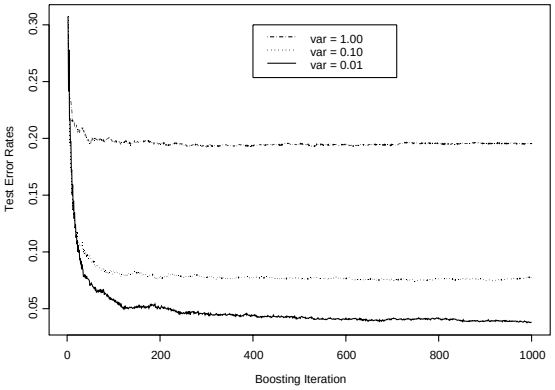


(d) Three methods

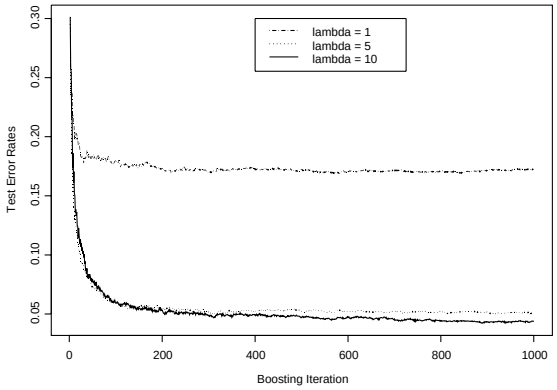
Figure 6: Comparison of three boosting methods for 2% mislabeled data near decision boundary. (a) Test error rates of AdaBoost. (b) Test error rates of SNRBoost. (c) Test error rates of MarginBoost.L_1. (d) Test error rates of three Boosting Methods.



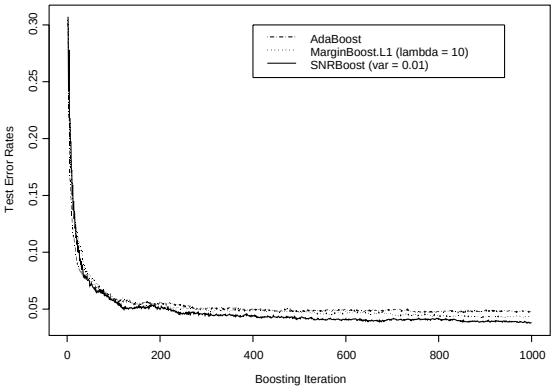
(a) AdaBoost



(b) SNRBoost



(c) MarginBoost.L_1



(d) Three methods

Figure 7: Comparison of three boosting methods for noiseless data. (a) Test error rates of AdaBoost. (b) Test error rates of SNRBoost. (c) Test error rates of MarginBoost.L_1. (d) Test error rates of three Boosting Methods.