INSTITUTE OF POLICY AND PLANNING SCIENCES

Discussion Paper Series

No. 1038

A Robust Boosting Method for Mislabeled Data

by

Natsuki Sano, Hideo Suzuki and Masato Koda

May 2003

UNIVERSITY OF TSUKUBA
Tsukuba, Ibaraki 305-8573
JAPAN

A Robust Boosting Method for Mislabeled Data

Natsuki Sano*, Hideo Suzuki†, and Masato Koda†

*The Doctoral Program in Policy and Planning Sciences, University of Tsukuba

†Institute of Policy and Planning Sciences, University of Tsukuba

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 in test error rates in the case of noisy, mislabeled situation.

Key words: AdaBoost, Machine Learning, Neural Network, Sigmoidal Function, Stagewise Additive Model.

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 Schapire (1990), Freund (1995), Freund and Schapire (1997)) 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." (Freund and Schapire, 1997). A concise description of AdaBoost is given here for the two-category classification setting. We have a set of N trainning 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, $F(x) = \sum_{m=1}^{M} \beta_m f_m(x)$, in which each $f_m(x)$ denotes a classifier producing values ± 1, and β_m are prescribed constants; the corresponding prediction (i.e., decision) is $\text{sign}(F(x))$.

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 desciption 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.

AdaBoost.M1. (Freund and Schapire, 1997)

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.

2. For $m = 1, 2, \ldots, M$ do:

   (a) Fit a classifier $f_m(x)$ to the training data using weights w_i.

   (b) Compute $\text{err}_m = \dfrac{\sum_{i=1}^{N} w_i I(y_i \neq f_m(x_i))}{\sum_{i=1}^{N} w_i}$.

   (c) Compute $\beta_m = \log((1 - \text{err}_m)/(\text{err}_m))$.

   (d) Set $w_i \leftarrow w_i \cdot \exp[\beta_m \cdot I(y_i \neq f_m(x_i))],\ i = 1, 2, \ldots, N$.

   end For

3. Output $F(x) = \text{sign}[\sum_{m=1}^{M} \beta_m f_m(x)]$.

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 called AdaBoost with trees as " the best off-the-shelf classifier in the world " (Breiman, 1998).

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 resistent to overfitting for low noise cases. However, recent studies with highly noisy patterns (Quinlan, 1996; Grove and Schuurmans, 1998; Rätsch et al., 1998) depict that it is clearly a myth that AdaBoost do not overfit since AdaBoost asymptotically concentrate on the patterns which are hardest to learn. To cope with this problem, some regularized Boosting algorithms (Mason et al., 1999; Rätsch et al., 1998) are proposed. In this paper, we will take an iterative improvement approach to optimize classifiers to derive a new robust Boosting method that is resistent 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

Friedman et al. (2000), and Hastie et al. (2001) 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 expasion coefficients, and basis functions $\{b(x; \gamma_m)\}_1^M$ 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 (2000)).

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,

$$\min_{\{\beta_m, \gamma_m\}_1^M} \sum_{i=1}^{N} L(y_i, \sum_{m=1}^{M} \beta_m b(x_i; \gamma_m)), \tag{2}$$

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 modeling

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 $\text{sign}(F_m(x))$.

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."

3 Derivation of Boosting Method Using Sigmoidal Loss Function

3.1 Sigmoidal Loss Function

Friedman et al. (2000) 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, $y \cdot F(x)$,

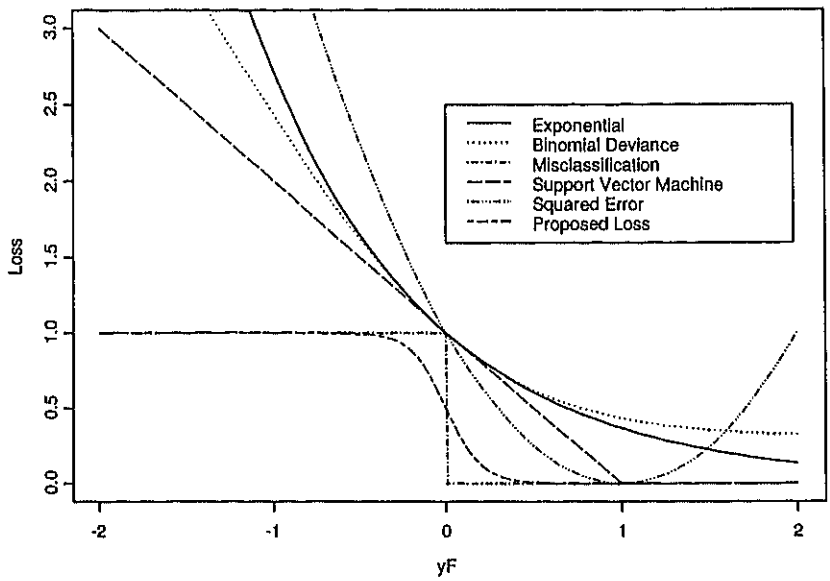


Figure 1: Loss functions for two-category classification

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, $L(y, F(x)) = I(y \cdot F(x) < 0)$, where $I(y \cdot F(x) < 0)$ denotes the indicator function with respect to the occurence 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 judgement on a zero-one loss function.

The exponential loss function exponentially penalizes negative margin obserbations 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 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 $y \cdot F(x) > 1$ incurrs 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. (2000) 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,

$$L(y, F(x)) = \frac{1}{1 + e^{\alpha y F(x)}}. \tag{3}$$

This is a sigmoidal function where α denotes the appropriate positive gain. Note that the proposed loss function (3) 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 optimizaton of Equation (3) by using the stegewise additive modelling can be formulated as follows:

$$\begin{aligned} (\beta_m, \gamma_m) &= \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \beta b(x_i; \gamma)), \\ &= \arg\min_{\beta,\gamma} \sum_{i=1}^{N} \frac{1}{1 + \exp(\alpha y_i (F_m(x_i) + \beta b(x_i; \gamma)))}, \end{aligned} \tag{4}$$

where $b(x; \gamma)$ denotes an appropriate base learner such as neural network. It should be noted that

the solution to Equation (4) 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 optimizaton involved in Equation (4) 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-ouput 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 (2001) for details of the approximation techniques.

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

$$L^N = \sum_{i=1}^{N} L(y_i, F(x_i)). \tag{5}$$

This empirical loss is well-defined and clearly is a function of the modelling parameters, i.e., β and γ. Let $\mathbf{F} = \{F(x_1), \ldots, F(x_i), \ldots, F(x_N)\}$ be considered as "parameters" to approximate the continuous function $F(x)$ at each of the N data points x_i. Then, minimization of Equation (3) 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

$$g_{im} = \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x_i)=F_{m-1}(x_i)}. \tag{6}$$

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 trainning samples. This task 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.

In order to fit negative gradients, $-\mathbf{g}_m$, method of least squares minimization is employed here. From Equation (6), we compute the components of negative gradient, $\tilde{y}_{im}$, utilizing the proposed loss function (3) as follows:

$$\tilde{y}_{im} = -g_{im} = -\left[\frac{\partial}{\partial F(x_i)}\frac{1}{1+e^{\alpha y_i F(x_i)}}\right]_{F(x_i)=F_{m-1}(x_i)} = \left.\frac{\alpha y_i e^{\alpha y_i F(x_i)}}{(1+e^{\alpha y_i F(x_i)})^2}\right|_{F(x_i)=F_{m-1}(x_i)}. \quad (7)$$

Using Equation (7), method of least squares minimization can be formulated as

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{N} [\tilde{y}_{im} - b(x_i;\gamma)]^2, \quad (8)$$

for an appropriate base learner $b(x;\gamma)$.

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

$$y_{im} = y_{i(m-1)} + \delta y_{im} = y_{i(m-1)} - \beta_m g_{im} = y_{i(m-1)} + \beta_m \tilde{y}_{im}, \quad (9)$$

where the " correction " term δy_{im} is generally a function of the input feature x_i, its class label, and the current state $y_{i(m-1)}$, 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 (9) can be generalized as

$$F_m(x) = F_{m-1}(x) + \beta_m b(x;\gamma_m), \quad (10)$$

where $b(x;\gamma_m)$ denotes the optimal base learner obtained from Equation (8).

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 $F_0(x) = \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$.

2. For $m = 1, 2, \ldots, M$ do:

   (a) $\tilde{y}_{im} = \left.\dfrac{\alpha y_i e^{\alpha y_i F(x_i)}}{(1 + e^{\alpha y_i F(x_i)})^2}\right|_{F(x_i)=F_{m-1}(x_i)}$

   (b) $\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{N} [\tilde{y}_{im} - b(x_i; \gamma)]^2$

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

   end For

3. Output $\text{sign}(F_m(x))$.

For the appropriate choice of β_m, the usual care and considerations associated with the gradient descent algorithm, i.e., Equation (9), apply here. Since the present Boosting method involves a gradient numerical search in order to generalize gradient features, the solution provided by the method may be a local minimum and not necessarily a global optimal one.

4 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. We focus our attention to classification results for mislabeled (i.e., noisy) and correctly labeled (i.e., noiseless) cases. The back-propagation neural network with single hidden layer is used as a base learner for both AdaBoost and the proposed method. Since a muiltilayer 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 offers the potential of ideal base learner for the experiments.

For numerical experiments, 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 $y = \text{sign}(F(\mathbf{x}))$, 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 20% 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 Figure 2(a) noiseless original data is shown while Figure 2(b) shows 2% mislabeled, noisy data. As easily observed, misslabeled data is scarce in the vicinity of the decision boundary.

The number of training and test data are 1000 each. The experiments of the noiseless case in which no mislabeled data is contained on the same dataset are also conducted. As for the neural network used as a base learner, the number of units in hidden layer (i.e., hidden units) takes the values, 3, 5, and 10, respectively. Note that the computational complexity of the present method can be scaled by the number of hidden units in the neural network. Iteration number of the weak learner, which is referred to as round (number), is 200. The gain parameter of the proposed loss function, α, is set to 1, and $\beta = 1$ is assumed for computational simplicity.

We plot in Figures 3-5 the learning (error minimization) curves for 2% mislabeled data. The number of hidden units is 3 for Figure 3, 5 for Figure 4, and 10 for Figure 5, respectively. In each figure, (a) shows AdaBoost training and test error rates, (b) training and test error rates of the proposed method, and (c) comparison of test error rates of AdaBoost and the proposed method. On test error rates, we may note that the proposed method is superior to AdaBoost except in the case where unit size (i.e., number of hidden units) is 3. The performance of the proposed method is better than that of AdaBoost as the number of hidden units increases. This trend may be accounted for in terms of the complexity of the base learner. Recall that the basic role played by the base learner in the proposed method is a superposition (generalization) of the gradients defined only at data points. Therefore, it is desirable that the complexity (i.e., unit size) of the weak learner is sufficiently large in order to fit and capture its gradient features.

Figures 6-8, on the other hand, illustrate the results for noiseless (0% mislabeled) cases, in which AdaBoost outperforms the proposed method on the average. Note that, however, the performance of the proposed method improves as the number of hidden units increases. Of particular significance is Figure 8(c), where the proposed method gradually catches up the performance of AdaBoost for the case with unit size 10.

Figures 9 and 10 compare the changes of test error rates in terms of the unit size of neural network in 2% mislabeled and error-free cases, respectively. In Figures 9 and 10, (a) shows AdaBoost, and (b) the proposed method. The peformance of the proposed method becomes better as the number of hidden units increases. On the other hand, the AdaBoost error is always fluctuating for both noisy and error-free cases and there is no clear trend of performance improvement with respect to the increase of unit size. Figure 9(a) indicates that the performance of the most complex AdaBoost with unit size 10 is worse for 2% mislabeled case. This implies that the complex learner may not be very appropriate for AdaBoost in noisy, mislabeled situation.

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 through intensive numerical experiments. The proposed method is robust compared to AdaBoost especially in mislabeled, noisy cases. In noiseless (0% mislabeled) cases, if the complex neural network (where the number of hidden units is large) is used as a base learner, similar performance equivalent to that of AdaBoost is attained for the proposed method. It should be noted that the performance of the method depends on the complexity of the base learner (i.e., neural network).

Acknowledgments

We thank Noboru Murata for useful comments on an earlier draft. 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

Breiman, L. (1998): Combining Predictors. Technical Report, Statistics Department, University of California, Berkeley.

Freund, Y. (1995): Boosting a weak learning algorithm by majority. *Information and Computation* **121** (2), 256-285.

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

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

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

Grove, A. and Schuurmans, D. (1998): Boosting in the limit: Maximizing the margin of learned ensembles. *Proc. 15th National Conference on Artifical Intelligence*, 692-699.

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

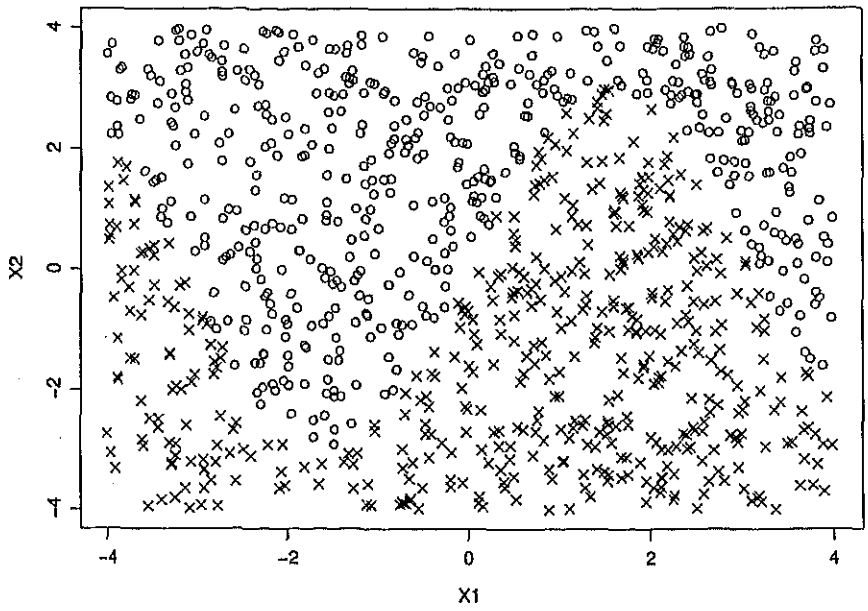
Koda, M. and Okano, H. (2000): A new stochastic learning algorithm for neural networks. *Journal of the Operations Research Society of Japan*, **43** (4), 469-485.

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

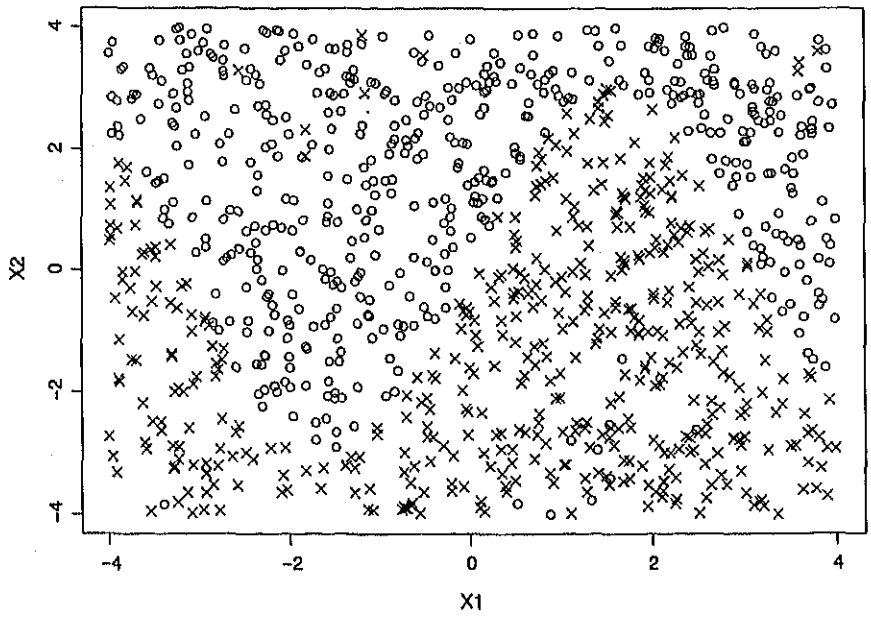
Quinlan, J. R. (1996): Boosting first-order learning. *Proc. 7th International Workshop on Algorithmic Learning Theory*, **1160**, 143-155.

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

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

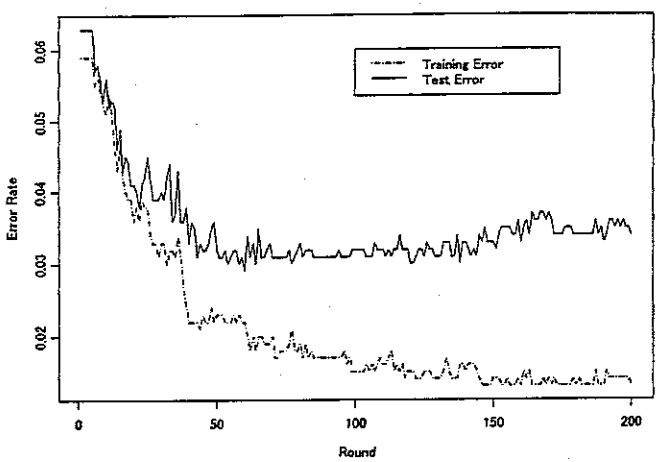


(a) Noiseless Original Data

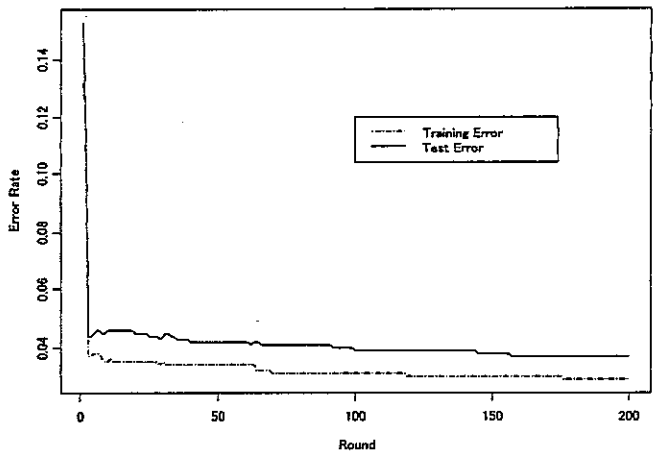


b) 2% Mislabeled Data

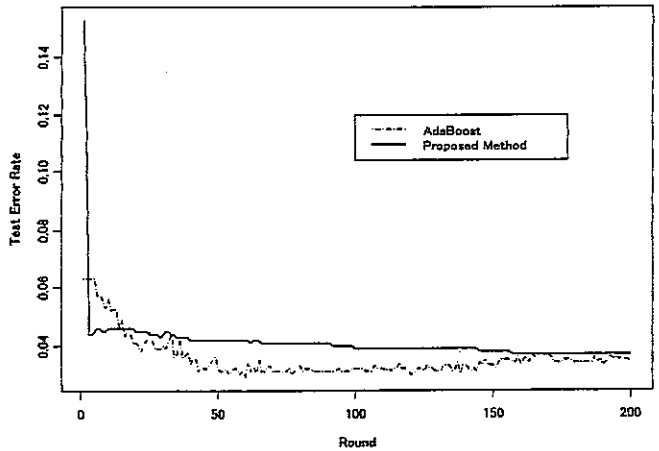
Figure 2. Training Data



(a) AdaBoost

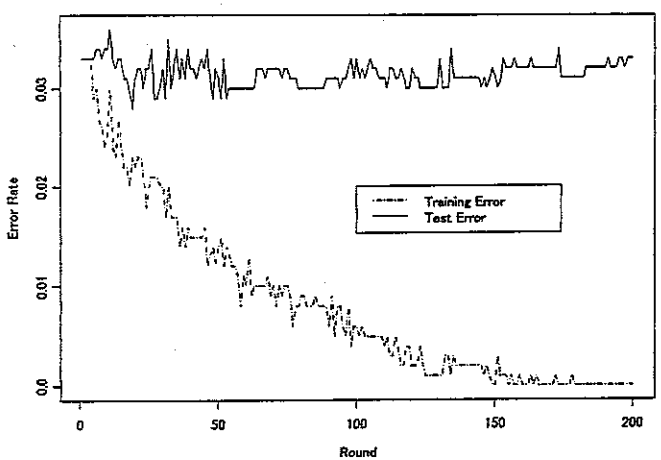


(b) Proposed Method

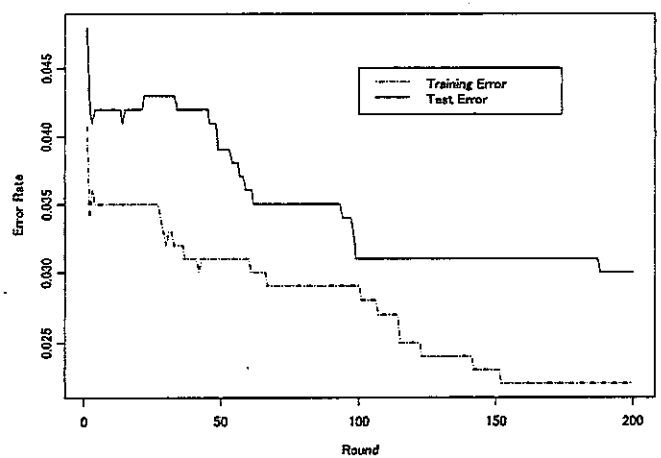


(c) Comparison of Test Errors

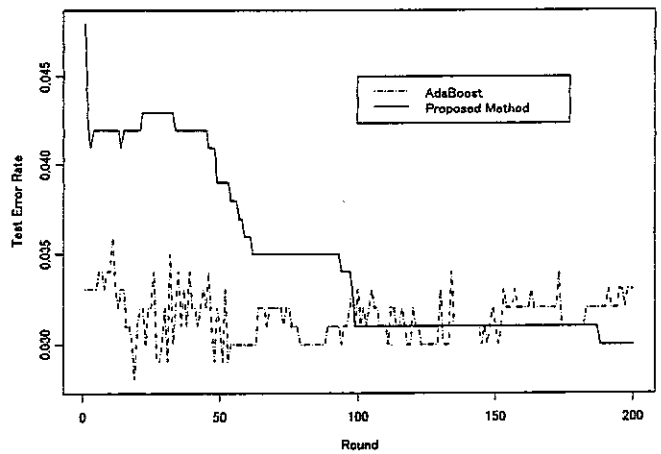
Figure 3. Case with the unit size of hidden layer is 3 for 2% mislabeled data



(a) AdaBoost

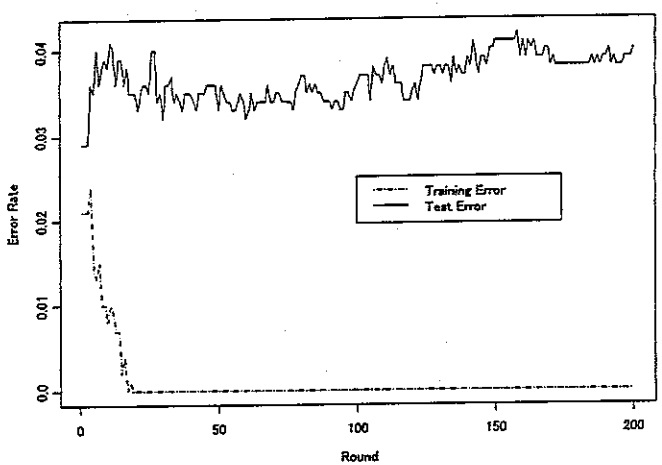


(b) Proposed Method

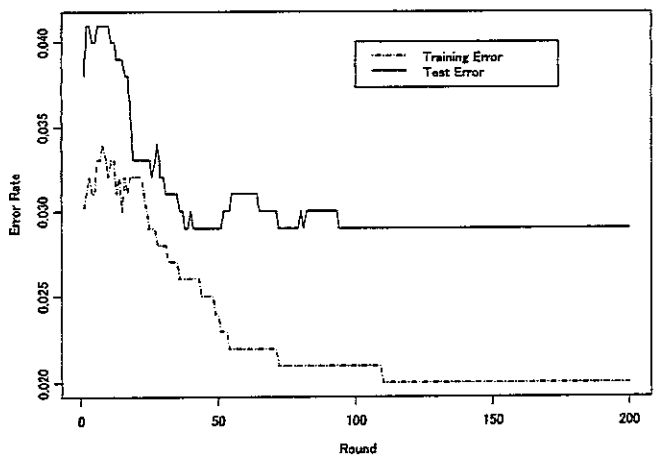


(c) Comparison of Test Errors

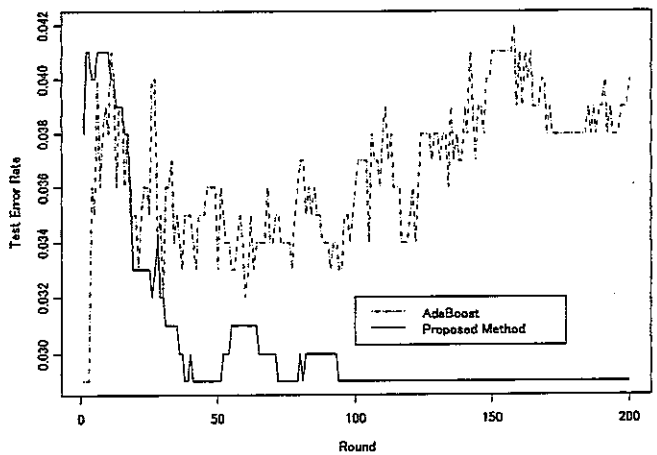
Figure 4. Case with the unit size of hidden layer is 5 for 2% mislabeled data



(a) AdaBoost

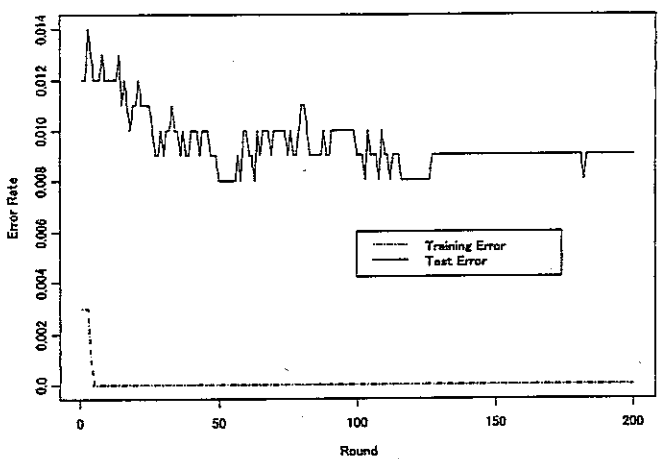


(b) Proposed Method

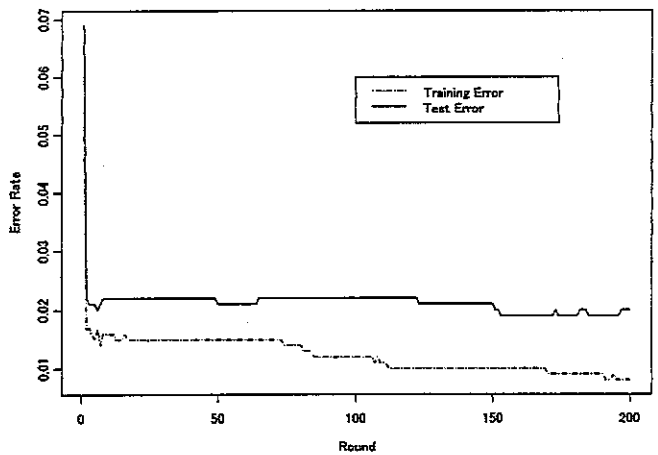


(c) Comparison of Test Errors

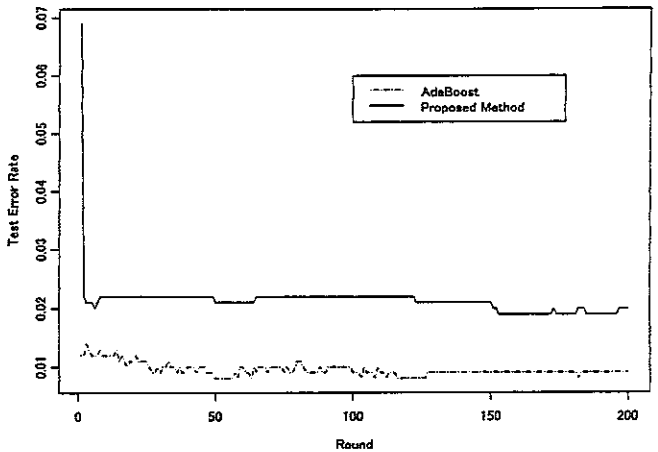
Figure 5. Case with the unit size of hidden layer is 10 for 2% mislabeled data



(a) AdaBoost

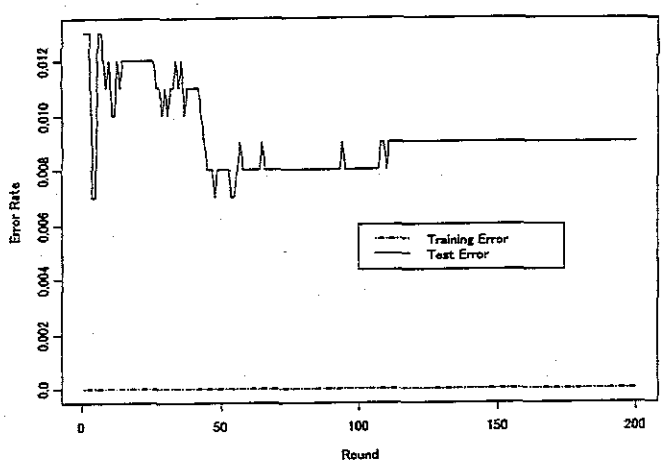


(b) Proposed Method

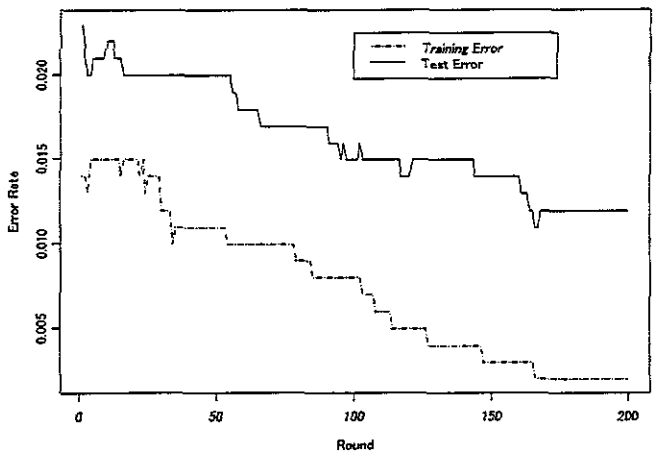


(c) Comparison of Test Errors

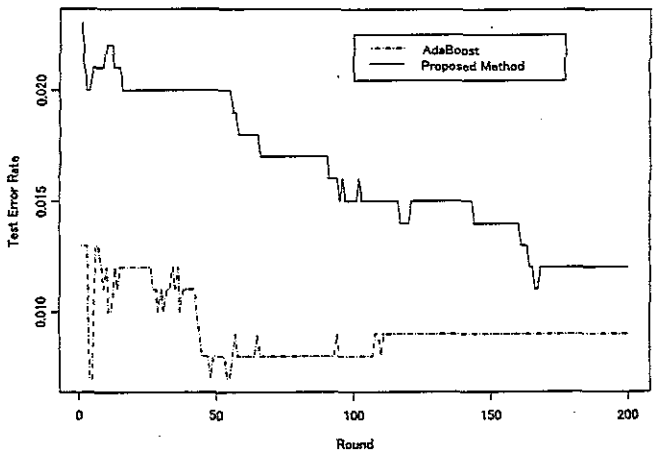
Figure 6. Case with the unit size of hidden layer is 3 for 0% mislabeled data



(a) AdaBoost

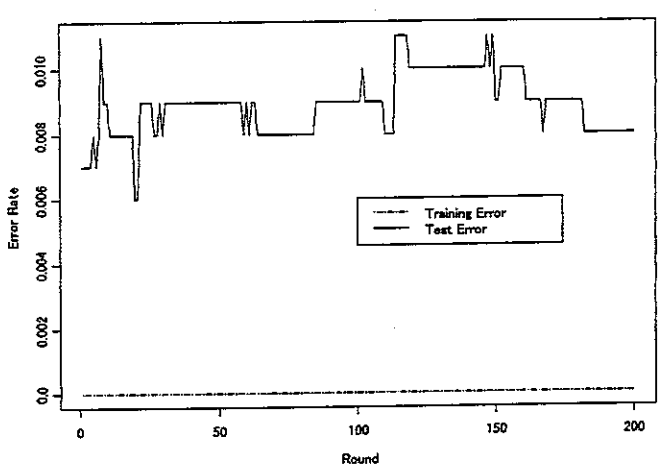


(b) Proposed Method

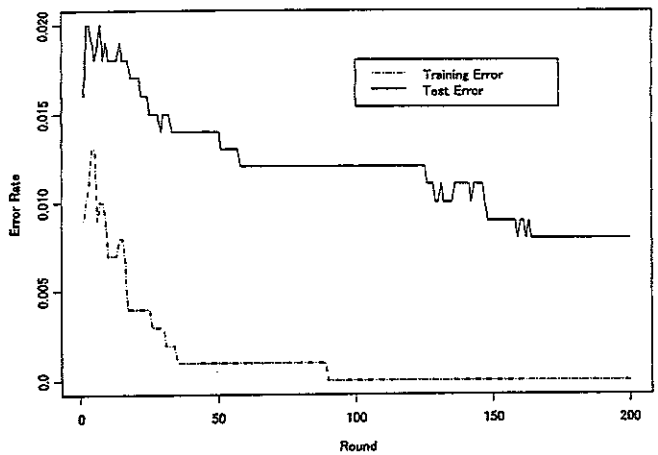


(c) Comparison of Test Errors

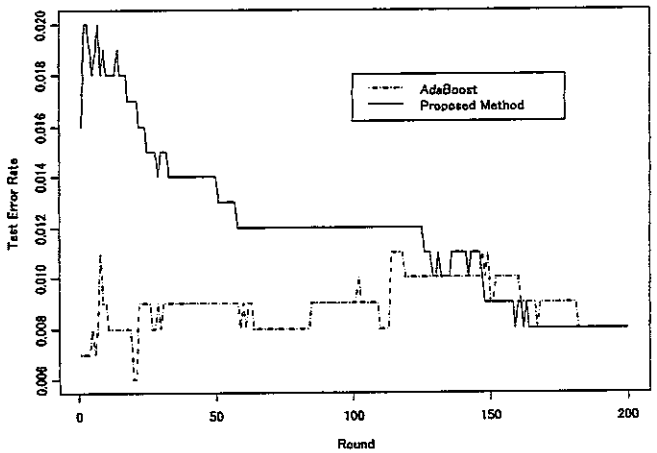
Figure 7. Case with the unit size of hidden layer is 5 for 0% mislabeled data



(a) AdaBoost

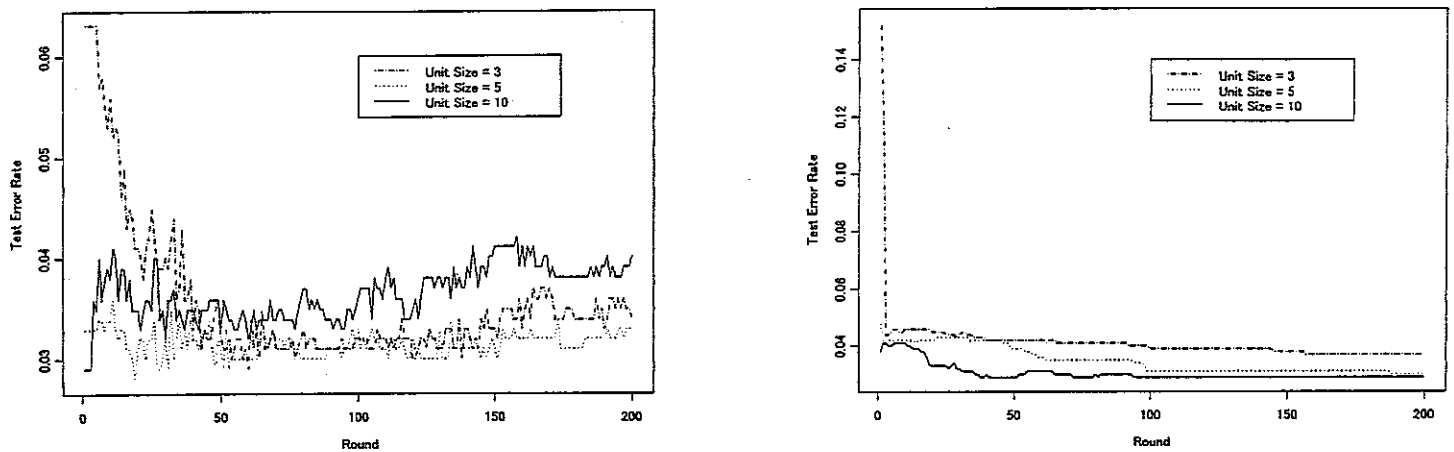


(b) Proposed Method



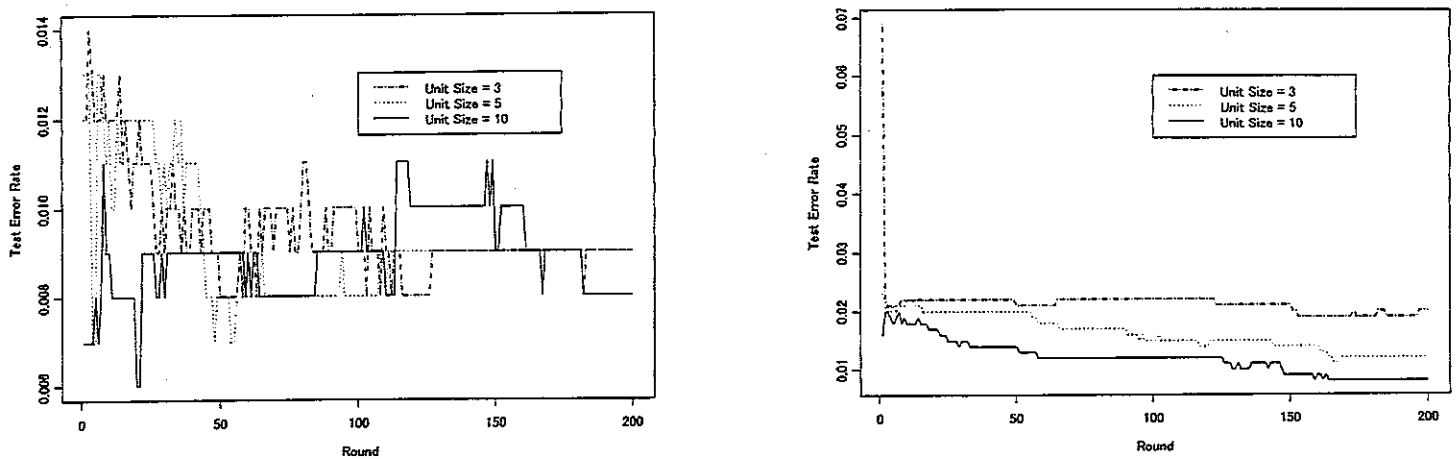
(c) Comparison of Test Errors

Figure 8. Case with the unit size of hidden layer is 10 for 0% mislabeled data



(a) AdaBoost (b) Proposed Method

Figure 9. Comparison of test errors with respect to the number of hidden units for 2% mislabeled data



(a) AdaBoost

(b) Proposed Method

Figure 10. Comparison of test errors with respect to the number of hidden units for 0% mislabeled data