IEICE TRANS. INF. & SYST., VOL.E98–D, NO.11 NOVEMBER 2015

PAPER

Penalized AdaBoost: Improving the Generalization Error of Gentle AdaBoost through a Margin Distribution

Shuqiong WU[†a)], *Nonmember and* **Hiroshi NAGAHASHI**[††b)], *Member*

SUMMARY Gentle AdaBoost is widely used in object detection and pattern recognition due to its efficiency and stability. To focus on instances with small margins, Gentle AdaBoost assigns larger weights to these instances during the training. However, misclassification of small-margin instances can still occur, which will cause the weights of these instances to become larger and larger. Eventually, several large-weight instances might dominate the whole data distribution, encouraging Gentle AdaBoost to choose weak hypotheses that fit only these instances in the late training phase. This phenomenon, known as "classifier distortion", degrades the generalization error and can easily lead to overfitting since the deviation of all selected weak hypotheses is increased by the late-selected ones. To solve this problem, we propose a new variant which we call "Penalized AdaBoost". In each iteration, our approach not only penalizes the misclassification of instances with small margins but also restrains the weight increase for instances with minimal margins. Our method performs better than Gentle AdaBoost because it avoids the "classifier distortion" effectively. Experiments show that our method achieves far lower generalization errors and a similar training speed compared with Gentle AdaBoost.

key words: *Gentle AdaBoost, generalization error, machine learning, modest AdaBoost, training error*

1. Introduction

AdaBoost was first introduced to machine learning researchers by Freund and Schapire [1], and has achieved considerable success in object detection [2]. It has been shown that AdaBoost is effective at enlarging the classification margins during the training process [3]. With the increased popularity of AdaBoost, a generalized version, Real AdaBoost, was proposed by Schapire and Singer [4]. Real AdaBoost applies confidence-prediction and domain-partitioning to the boosting process. In 2000, Friedman et al. analyzed AdaBoost from the viewpoint of an additive logistic model and devised Gentle AdaBoost, which optimizes the training error by Newton steps. It has been shown that Gentle AdaBoost is more robust and stable than Real AdaBoost with respect to the generalization error [5]. Recently, many AdaBoost variants were proposed for various purposes. Some researchers aimed at speeding up the training process. For example, AdaTree was proposed to improve the training speed by combining its weak classifiers non-linearly [6]. In addition, several approaches were also introduced to speed up the training phase [7]–[9]. They trade off the integrity of training data for faster training.

Some AdaBoost variants were devised to improve the robustness against noise data. For instance, BrownBoost was proposed to reduce the influence of outliers in training data [10]. The review work showed that BrownBoost performs similarly to Gentle AdaBoost [11]. MadaBoost and SmoothBoost were introduced to improve the robustness against malicious noise [12], [13]. Nevertheless, MadaBoost requires the error rates of its weak hypotheses to be increased monotonically, and SmoothBoost requires noise tolerant weak hypotheses [14]. Similarly, regularized AdaBoost was developed to solve overfitting problems caused by noise instances. Unfortunately, it needs validation subsets to identify and regulate the overfitting iteratively [15], [16]. For the same purpose, Freund has proposed RobustBoost which is an extension of BrownBoost [17].

Other AdaBoost variants were devised to improve the generalization ability such as SemiBoost and FloatBoost. SemiBoost combines supervised learning with semi-supervised learning by utilizing both the labelled and unlabelled training data [18]. Its purpose is to reduce the generalization error when the labelled training data are insufficient [19]. FloatBoost deletes the less effective weak classifiers during the training by a backtracking mechanism so that it outperforms AdaBoost when it has the same number of weak classifiers as AdaBoost [20]. However, FloatBoost requires far more training cycles than AdaBoost. Similarly, LPBoost was introduced to suppress the generalization error of AdaBoost [21]. It uses linear programming to minimize the minimal margin directly [21]. However, a comparison indicated that LPBoost generally performs worse than AdaBoost [22]. Later, Vezhnevets and Vezhnevets proposed Modest AdaBoost to reduce the generalization error of Gentle AdaBoost [23]. It occasionally performs better than Gentle AdaBoost by strengthening the contribution of weak hypotheses which work well on instances with small margins [24]. However, it is difficult to decrease the training error of Modest AdaBoost. Moreover, its performance is unstable since the accuracy drops in some data sets. Interactive Boosting, ReweightBoost, and SoftBoost were also devised to decrease the generalization error [25]–[27]. Interactive Boosting not only assigns weights to training data but also gives weights to features [25]. Reweight-

Manuscript received February 18, 2015.
Manuscript revised July 12, 2015.
Manuscript publicized August 13, 2015.

[†]The author is with the Department of Computational Intelligence and Systems Science, Tokyo Institute of Technology, Yokohama-shi, 226–8503 Japan.
[††]The author is with the Imaging Science and Engineering Laboratory, Tokyo Institute of Technology, Yokohama-shi, 226–8503 Japan.
a) E-mail: bu.s.aa@m.titech.ac.jp
b) E-mail: longb@isl.titech.ac.jp

DOI: 10.1587/transinf.2015EDP7069

Boost reuses the selected weak classifiers in a tree structure, but it restricts its weak classifiers to stump decision trees [26]. SoftBoost optimizes a soft margin instead of the hard margin used in AdaBoost [27]. An extended research of [27] combines SoftBoost with LPBoost [28]. All the four AdaBoost variants have achieved better performance than AdaBoost [29]. However, they introduced complicated calculations which may lead to a longer training time. In 2014, another method called Margin-pruning Boost was also created to decrease the generalization error of Gentle AdaBoost [30]. It achieved lower generalization errors than Gentle AdaBoost by correcting "classifier distortion" in each loop. However, its performance drops as the number of iterations increases.

In this paper, we analyze the problem of Margin-pruning Boost, and improve it by using an adaptive resetting technique. Furthermore, we introduce a penalty policy to restrain the reduction of small margins. Because the proposed approach enlarges the margins more than Gentle AdaBoost and Margin-pruning Boost, it can obtain better performance than the two variants. Moreover, the calculation of our method is simple, and it can be easily applied to other variants such as regularized AdaBoost, Interactive Boosting, ReweightBoost, FloatBoost, SemiBoost, RobustBoost, and so on. The remainder of this paper is organized as follows. Section 2 briefly introduces Gentle AdaBoost and Margin-pruning Boost. Section 3 explains our proposed Penalized AdaBoost, and then analyzes its generalization ability in terms of the classification margins. Section 4 shows the experimental results, Sect. 5 discusses the proposed algorithm, and Sect. 6 concludes our work.

2. Related Work

This section explains Gentle AdaBoost and our previous work Margin-pruning Boost in details. Here let $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_i, y_i), \ldots, (x_N, y_N)\}$ be a training set that contains N training instances. For each instance x_i, if it is a positive instance, y_i equals to 1; otherwise, y_i equals to -1. Here $w_{i,t}$ is a weight assigned to the instance x_i in the t-th iteration. In this paper, we focus on binary classification problems. There are many types of weak classifiers such as SVM, CART, Decision tree, and so on. Nevertheless, we use stump decision trees since they are the simplest classifiers which can effectively show the differences between boosting algorithms [23].

2.1 Gentle AdaBoost

Gentle AdaBoost is a variant of AdaBoost, it calculates one weak hypothesis in each iteration, and finally combines these weak hypotheses in a linear manner. Gentle AdaBoost calculates the weak hypothesis by optimizing the weighted least square error in each run [5]. Being the same as AdaBoost, Gentle AdaBoost increases weights for misclassified instances exponentially. Thus, it can easily focus on the difficult-to-classify instances and try to correctly classify them. However, these difficult-to-classify instances may be noise or outliers. In this case, the weight of these instances will be increased again and again. Finally several large-weight instances force the late selected weak classifiers to fit them only rather than fit the majority instances. This phenomenon called "classifier distortion" deteriorates the generalization errors.

2.2 Classification Margin

The classification margin (also called margin) of a given instance x_i is defined as the difference between the prediction confidence of weak hypotheses leading to correct classification and that of weak hypotheses leading to misclassification [3]. It is a real number in the range $[-1, 1]$, and the instance x_i is correctly classified if and only if its margin is positive [3]. Thus, the margin of an instance x_i in AdaBoost is defined as follows [3]:

$$\text{Margin}_T(x_i) = y_i \sum_{t=1}^{T} \alpha_t h_t(x_i) / \sum_{t=1}^{T} \alpha_t. \quad (1)$$

Here we suppose AdaBoost combines T weak classifiers after T iterations. Thereby, $\sum_{t=1}^{T} \alpha_t h_t(x_i)$ in (1) means the strong classifier [1], and $h_t(x_i)$ denotes the weak classifier at the t-th iteration which is also a function from the instance space X to {-1,+1}.

In Gentle AdaBoost, confidence-prediction is applied to the boosting process by folding α_t into the weak hypothesis, i.e., for an instance x_i, the weak hypothesis at the t-th iteration $f_t(x_i)$ is equivalent to $\alpha_t h_t(x_i)$ of AdaBoost [4], [5]. Here the sign of the weak hypothesis $f_t(x_i)$ represents the class of the weighted majority of training data classified into the same partition and the absolute value $|f_t(x_i)|$ shows the prediction confidence [4], [5]. The larger the prediction confidence is, the higher classification ability $f_t(x_i)$ has. Differently from the weak hypothesis $h_t(x_i)$ defined in AdaBoost, here the weak hypothesis $f_t(x_i)$ is a function from instance space X to real numbers. The prediction confidence $|f_t(x_i)|$ is equivalent to α_t in (1). From (1), the margin in Gentle AdaBoost can be defined as follows:

$$\text{Margin}_T(x_i) = y_i \sum_{t=1}^{T} f_t(x_i) \; / \sum_{t=1}^{T} |f_t(x_i)|. \quad (2)$$

In (2), $f_t(x_i)$ refers to the weak hypothesis for instance x_i at the t-th iteration, where T is the number of iterations. Thus, $\sum_{t=1}^{T} f_t(x_i)$ denotes the strong hypothesis after T iterations. Since we apply confidence-prediction into Margin-pruning Boost and our proposed Penalized AdaBoost, (2) is also suitable for Margin-pruning Boost and Penalized AdaBoost.

2.3 Margin-Pruning Boost

To avoid the classifier distortion of Gentle AdaBoost, we proposed Margin-pruning Boost. Margin-pruning Boost is explained as follows [30]:

Algorithm 1. Margin-pruning Boost

1. Set the initial weight $w_{i,1} = 1/N$, where $i = 1, 2, \ldots, N$.

2. For $t = 1, 2, \ldots, T$, run the following steps:
(a) Train a stump tree that classifies all training data into a positive set S_t^1 and a negative set S_t^2. For each set S_t^j where $j \in \{1, 2\}$, calculate W_{t+}^j and W_{t-}^j as follows:

$$W_{t+}^j = \sum\nolimits_{i:x_i \in S_t^j \wedge y_i = 1} w_{i,t}. \tag{3}$$

$$W_{t-}^j = \sum\nolimits_{i:x_i \in S_t^j \wedge y_i = -1} w_{i,t}. \tag{4}$$

(b) Compute weak hypotheses for $j \in \{1, 2\}$ as follows:

$$f_t^j(x) = (W_{t+}^j - W_{t-}^j)/(W_{t+}^j + W_{t-}^j), \tag{5}$$

For each training instance x_i, let its weak hypothesis $f_t(x_i)$ be $f_t^1(x)$ if it is classified as a positive instance by the stump tree; otherwise set $f_t(x_i) = f_t^2(x)$.
(c) Update all instance weights by

$$w_{i,t+1} = exp[-y_i \times \sum\nolimits_{p=1}^{t} f_p(x_i)]. \tag{6}$$

(d) For each instance x_i, if $w_{i,t+1} > Q_{t+1}$, reset its weight and the sum of weak hypotheses by

$$w_{i,t+1} = 1, \quad \sum\nolimits_{p=1}^{t} f_p(x_i) = 0. \tag{7}$$

$$Q_{t+1} = max_i\{w_{i,t+1}\} - \frac{max_i\{w_{i,t+1}\} - min_i\{w_{i,t+1}\}}{\beta}. \tag{8}$$

Then let $w_{i,t+1} = w_{i,t+1}/Z_t$, where Z_t equals to $\sum_i w_{i,t+1}$.
3. Set $F_T(x) = \sum_{t=1}^{T} f_t(x)$, and output the strong classifier $H(x) = sign[F_T(x)]$.

Compared with Gentle AdaBoost, Margin-pruning Boost added Step 2(d). In Step 2(d), the parameter β is tuned by experiments [30]. In Margin-pruning Boost, instances whose weights exceed the threshold at the current loop are regarded as noise-like data. Thus, the sum of the weak hypotheses for these instances is reset to be 0. This means that the current strong hypothesis can not classify these instances. Accordingly, the weights of these noise-like instances are also reset to 1 to limit their increase. However, instances exceed the threshold are not necessarily noise. Thus, Margin-pruning Boost still computes weak hypotheses for these instances after resetting because it still tries to correctly classify them in future loops. The purpose of resetting is to keep the weights of noise-like instances small to avoid classifier distortion. Nevertheless, the resetting works at the early training phase but fails at the late training phase. At the early training phase, the weights of instances filtered by the thresholding are almost large than 1. Thereby, resetting the weights of these instances to be 1 can reduce the importance of these instances. However, as the number of iterations increases, the weights of filtered instances are probably smaller than 1. In this case, the resetting fails to avoid classifier distortion.

3. Proposed Algorithm

3.1 Penalized AdaBoost

To solve the problem of Margin-pruning Boost we discussed above, we propose a new method called "Penalized AdaBoost" in this section. Differently from Margin-pruning Boost, only the instances whose weights exceed the threshold and margins are negative are reset. From (2) and (6), we can see that instances whose margins are negative always have a weight larger than 1. Therefore, resetting the weights of these filtered instances by the thresholding to be 1 on the premise that the margins of these instances are negative can always reduces the importance of these instances even when the number of iterations increases. We can see this point in Step 2(e) of the following algorithm. In addition, compared with Gentle AdaBoost and Margin-pruning Boost, we use a margin distribution from the previous loop to restrict the misclassification of instances with small margins in the current loop in our proposed algorithm (shown in Step 2(b) and 2(c) of the following algorithm). This improvement reduces the confidence prediction of weak hypotheses that misclassify small-margin instances. Thus, it can enlarge the margins more than Gentle AdaBoost and Margin-pruning Boost. We will explain this point in more detail in Sect. 3.2 by margin distributions. Next we describe the new method by the following pseudocode :

Algorithm 2. Penalized AdaBoost

1. Set the initial weight $w_{i,1} = 1/N$, where $i = 1, 2, \ldots, N$.
2. For $t = 1, 2, \ldots, T$, run the following steps:
(a) Train a stump tree based on the weighted training instances, and then calculate W_{t+}^j and W_{t-}^j for each set S_t^j ($j \in \{1, 2\}$) the same as in Step 2(a) of Algorithm 1.
(b) Calculate a margin feedback factor $m_{i,t}$ as shown in (9) for $t > 1$, and set $m_{i,1} = 1/N$ for $t = 1$.

$$m_{i,t} = \exp(-\text{Margin}_{t-1}(x_i))/U_t. \tag{9}$$

In (9), U_t equals to $\sum_i \exp(-\text{Margin}_{t-1}(x_i))$. Then compute M_{t+}^j and M_{t-}^j ($j \in \{1, 2\}$) for each set S_t^j as

$$M_{t+}^j = \sum\nolimits_{i:x_i \in S_t^j \wedge y_i = 1} m_{i,t}. \tag{10}$$

$$M_{t-}^j = \sum\nolimits_{i:x_i \in S_t^j \wedge y_i = -1} m_{i,t}. \tag{11}$$

(c) Compute weak hypotheses for $j \in \{1, 2\}$ as follows:

$$f_t^j(x) = \begin{cases} (W_{t+}^j - W_{t-}^j)(1 - M_{t-}^j), & if\ W_{t+}^j > W_{t-}^j \\ (W_{t+}^j - W_{t-}^j)(1 - M_{t+}^j), & otherwise \end{cases}. \tag{12}$$

For each training instance x_i, let its weak hypothesis $f_t(x_i)$ be $f_t^1(x)$ if it is classified as a positive instance by the stump tree; otherwise set $f_t(x_i) = f_t^2(x)$.
(d) Update all instance weights by (6).
(e) For each instance x_i, if $w_{i,t+1} > Q_{t+1}$ and $\text{Margin}_t(x_i) < 0$, reset its weight and the sum of weak hypotheses by

$$w_{i,t+1} = 1, \quad \sum\nolimits_{p=1}^{t} f_p(x_i) = 0. \tag{13}$$

$$Q_{t+1} = max_i\{w_{i,t+1}\} - \frac{max_i\{w_{i,t+1}\} - min_i\{w_{i,t+1}\}}{\gamma}. \tag{14}$$

Then let $w_{i,t+1} = w_{i,t+1}/Z_t$, where Z_t equals to $\sum_i w_{i,t+1}$.
3. Set $F_T(x) = \sum_{t=1}^{T} f_t(x)$, and output the strong classifier $H(x) = sign[F_T(x)]$.

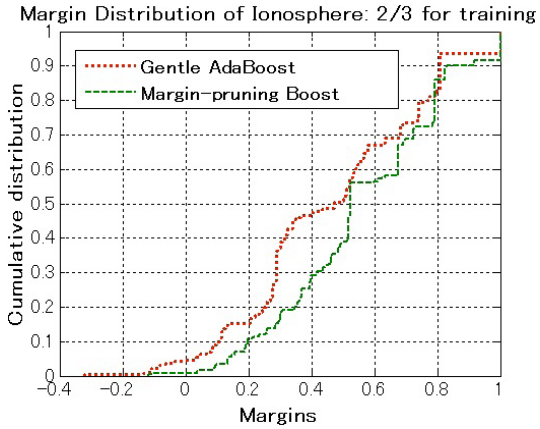
In (13), for an instance x_i, if its weight exceeds the threshold and its margin is negative, this means that instance x_i may be noise data since it is misclassified many times from Iteration 1 to Iteration t. Thereby we reset the summed weak hypotheses $\sum_{p=1}^{t} f_p(x_i)$ to be 0 since the current summed weak hypotheses can not correctly classify instance x_i. However, instance x_i is not necessarily noise. Thus Penalized AdaBoost still calculates the weak hypotheses for instance x_i after its resetting. Although the summed weak hypotheses are reset at the t-th iteration, there is still a chance to correctly classify instance x_i after the t-th iteration. This kind of policy restrains the weight increase for these noise-like instances, but still tries to correctly classify these instances.

3.2 Statistical Analysis for Improvement

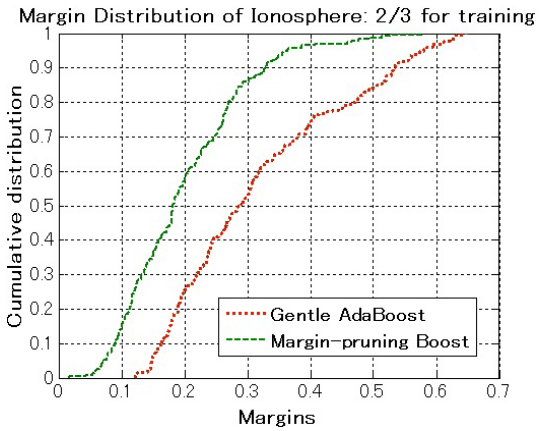
In this section, we analyze how Penalized AdaBoost restrains the misclassification of instances with small margins, and how the improved resetting technique enlarges the margins more than Gentle AdaBoost and Margin-pruning Boost.

In Gentle AdaBoost, if an instance x_i is misclassified by the current weak hypothesis, its weight will be increased exponentially no matter how large or small its margin is. Accordingly, its margin will be reduced due to this misclassification. The goal of the weight adjustment policy in Gentle AdaBoost is to make the future weak hypotheses focus more on instances with small margins and try to correctly classify them. However, misclassification of instances with small margins can still occur during the training phase especially when the data set contains some noise. In this case, the margins of these instances will become smaller so that their weights will become larger. Finally Gentle AdaBoost will select weak hypotheses to classify these minimal-margin instances because their weights are increased large sufficiently. Nevertheless, these selected weak hypotheses in the late training phase are more likely to fit the large-weight instances only, so that the performance of the strong hypothesis on other instances will be degraded. This kind of classifier distortion deteriorates the generalization error of the strong hypothesis since the deviation of all the selected weak hypotheses is increased in the late training as Gentle AdaBoost tries to fit instances with minimal margins. It has been demonstrated that the minimal margin of training data does not influence the generalization error [22]. However, the whole margin distribution which can obtain a balance between the training error and complexity is important to the generalization error [22]. Therefore, enlarging the margins of the whole data set is more important than fitting the minimal-margin instances.

In Margin-pruning Boost, we reset the weak hypotheses and weights of instances whose weights are larger than a threshold [30]. From (2) and (7), we can see that the margins of instances exceed the threshold are reset to be 0. In the early training phase, most of instances filtered by the thresholding have negative margins. In this case, resetting their margins to 0 can enlarge the whole margin distribution. This means the already selected weak hypotheses can not correctly classify these filtered instances so that the algorithm resets them to 0. After resetting, these instances are more likely to be correctly classified in the future runs since the influence of previous misclassification is removed. Consequently, the weights of these instances are reset to be smaller than their original weights to avoid "classifier distortion". However, as the number of iterations increases, instances filtered by the thresholding do not have negative margins necessarily. This means they maybe have positive margins, which indicate that they may be correctly classified by the combination of the currently selected weak hypotheses. Therefore, resetting these positive margins to be 0 will decrease the margins and degrade the classification performance. We can also see this point from the following margin distributions. Here margin distribution describes the probability that a training instance has a margin less than or equals to a given value X, the probability is described by Y-axis and the value X is described by X-axis. Figures 1 (a) and 1 (b) show the margin distributions of a data set Ionosphere at Iteration 10 and Iteration 100 respectively. From Fig. 1,



(a) Cumulative margin distribution at Iteration 10



(b) Cumulative margin distribution at Iteration 100

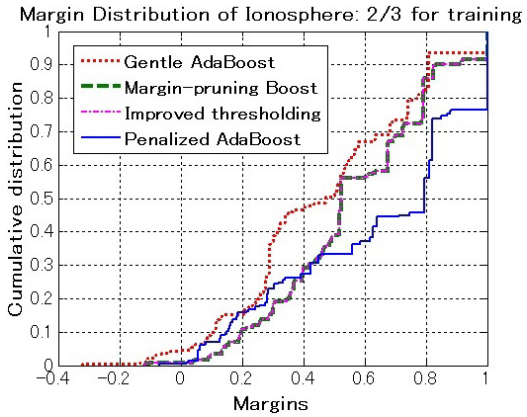
Fig. 1 Cumulative margin distributions at Iteration 10 and 100.

we can see that Margin-pruning Boost performs better than Gentle AdaBoost in the early training phase. However, the performance drops as the number of iterations increases.

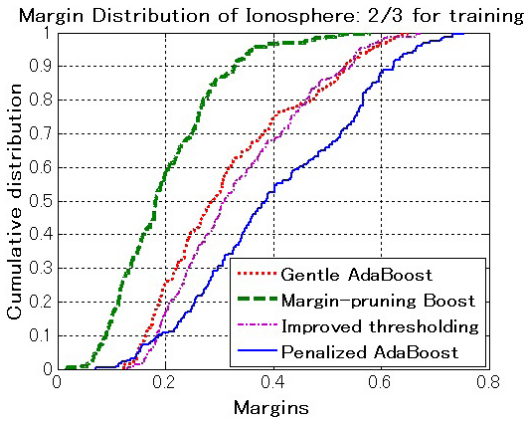
Compared with Gentle AdaBoost and Margin-pruning Boost, the new proposed Penalized AdaBoost has two advantages. Firstly it utilizes an improved thresholding by adding the condition that the margin must be negative for the reset instances. Compared with Margin-pruning Boost, only instances whose margins are negative and whose weights exceed the threshold are reset in Penalized AdaBoost. This means that the algorithm exactly reduces the importance of these filtered instances. On the other hand, it also removes the influence of weak hypotheses which give wrong predictions for these filtered instances. Thus, Penalized AdaBoost outperforms Margin-pruning Boost.

Secondly, Penalized AdaBoost introduces a penalty policy which uses the margins in the previous run to restrict the reduction of small margins. Since the penalty policy restrains the decrease of small margins, it can enlarge the margins more than Gentle AdaBoost, as we will demonstrate below. In Step 2(c) of Penalized AdaBoost, for a given set S_t^j, if $W_{t+}^j > W_{t-}^j$ holds, we can see that the sign of the current weak hypothesis $f_t^j(x)$ is +1. This means S_t^j is a positive set. Thus, instances classified into Set S_t^j and whose class is -1 are misclassified by $f_t^j(x)$. From (12), we notice that M_{t-}^j in the positive set is the sum of margin feedback factor $m_{i,t}$ of misclassified instances. Then from (9), we can see that the smaller the margin of the misclassified instance is, the larger $m_{i,t}$ is. Then according to (11), we can deduce that the larger $m_{i,t}$ is, the larger M_{t-}^j is, and the smaller $(1 - M_{t-}^j)$ is. This means that a misclassification of instances with small margins will lead to a small $(1 - M_{t-}^j)$, or we can say that $(1 - M_{t-}^j)$ is proportional to the margins. The small $(1 - M_{t-}^j)$ degrades the prediction confidence of the current weak hypothesis $f_t^j(x)$ so that the misclassification of these instances is easier to be corrected in future runs, compared with the case without $(1 - M_{t-}^j)$ in Gentle AdaBoost. Accordingly, the margin reduction of instances with small margins is also restricted by the small $(1-M_{t-}^j)$. By contrast, a misclassification of instances with large margins leads to a large $(1-M_{t-}^j)$. However, the influence of this misclassification is small, as these instances already have large margins. Although their weights are increased in the current iteration, they can not contribute to the "classifier distortion" because they were small in the previous run. If $W_{t+}^j \leq W_{t-}^j$ holds, the misclassification of instances with small margins is also penalized by reducing the prediction confidence of the weak hypothesis $f_t^j(x)$. In general, the penalty policy tries to achieve a best balance of the margin distribution.

Figure 2 shows the margin distributions of the data set Ionosphere, and Fig. 3 shows the generalization errors of the same data set. From Fig. 2 (a), we noticed that the improved thresholding (first advantage) of Penalized AdaBoost performs the same as Margin-pruning Boost when the number of iterations is small. However, it outperforms Margin-pruning Boost as the number of iterations increases (shown in Fig. 2 (b)). This means the improved thresholding exactly solves the problem of Margin-pruning Boost. This point can be also seen in Fig. 3 where Improved thresholding has better generalization errors than Margin-pruning Boost even when the number of iterations increases. When we see the curves of Penalized AdaBoost in Fig. 2 and Fig. 3, we find combining the improved thresholding and the penalty policy (second advantage) improves the performance more than the



(a) Cumulative margin distribution at Iteration 10



(b) Cumulative margin distribution at Iteration 100

Fig. 2 Cumulative margin distributions at Iteration 10 and 100.

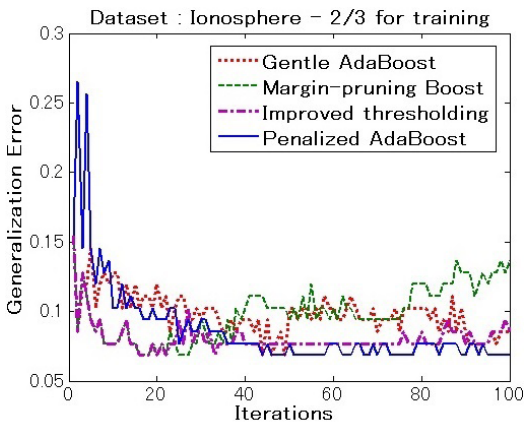


Fig. 3 Generalization errors of Ionosphere.

case only using the improved thresholding. Therefore, Penalized AdaBoost enlarges the margins more than Gentle AdaBoost and Margin-pruning Boost especially when the number of iterations increases. From Fig. 3, we can see that Penalized AdaBoost obtains a best generalization error as the number of iterations increases.

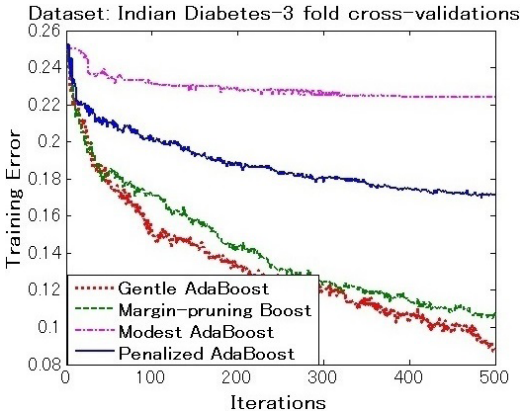
With respect to the parameter γ in (14) in Penalized AdaBoost, we measured the classification performance on 5 data sets (which are different from the tested data sets in our experiments) with different values of γ ($\gamma = 10, 30, 50, 70, 90$), and evaluated the overall performance on the 5 data sets. We found $\gamma = 50$ has the best generalization errors. So we use $\gamma = 50$ in Penalized AdaBoost. We used the same method to choose the parameter β in Margin-pruning Boost, and $\beta = 50$ outperforms other four values in Margin-pruning Boost. Thereby, we also set $\beta = 50$ in Margin-pruning Boost [30].

4. Experiments

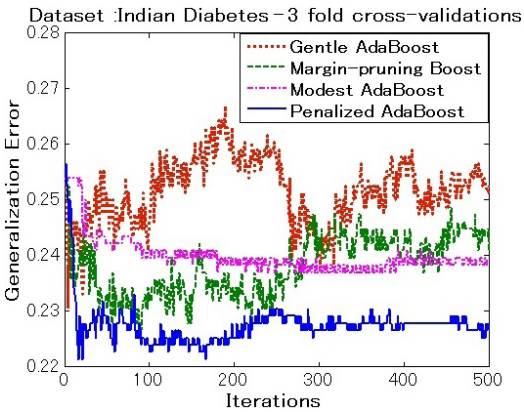
In this section, we compared our algorithm with Gentle AdaBoost [5], Modest AdaBoost [23], and Margin-pruning Boost [30] for 26 binary classification tasks from UCI repository [31]. These data sets can be downloaded from the UCI web page[†]. For each data set, we used 3-fold cross-validation to evaluate the training and generalization errors. We utilized Matlab AdaBoost Toolbox which provides the source code of many AdaBoost variants such as Gentle AdaBoost, Modest AdaBoost and so on [32]. The Matlab AdaBoost Toolbox (open-source) can be downloaded from the homepage of Graphics and Media Lab[††].

Figures 4 (a) and 4 (b) show the comparison of training and generalization errors for a data set Indian Diabetes. From Fig. 4, we can see that our method outperforms other three methods in terms of the generalization error, which we estimate by the classification error on the test set. Differently from Modest AdaBoost, the training error in our method is decreased gradually. Table 1 summarizes the results from all data sets. The generalization errors were measured after 200 iterations. We also measured the generalization errors in different data sets after 500 iterations. Table 2 shows the comparison results. In Table 1 and Table 2, No.S and No.F mean the number of instances and number of features, G-AB, M-AB, M-PB, and P-AB mean Gentle AdaBoost, Modest AdaBoost, Margin-pruning Boost and Penalized AdaBoost; and the bold numbers indicates the best performance. From Table 1 and Table 2, we noticed that Penalized AdaBoost overall outperforms other three methods. It achieves the best performance on 10 data sets in Table 1, and 16 data sets in Table 2. Comparing Table 1 with Table 2, we can also conclude that Penalized and Modest AdaBoost are more resistant to overfitting than Gentle AdaBoost and Margin-pruning Boost.

[†]http://archive.ics.uci.edu/ml/datasets.html
[††]http://graphics.cs.msu.ru/en/science/research/machinelearning



(a) Training errors of Indian Diabetes



(b) Generalization errors of Indian Diabetes

Fig. 4 (a) Training errors, and (b) Generalization errors.

5. Discussion

Gentle AdaBoost emphasizes the difficult-to-classify instances so that it can decrease the error rate rapidly in the training. Focusing on difficult-to-classify instances actually improves the classification performance. However, focusing too much on these instances may lead to overfitting. LP-Boost which directly optimizes the minimal margin is a typical example. It emphasizes the most difficult instances more than Gentle AdaBoost. Thereby, the selected weak classifiers are more likely to fit these most difficult instances instead of the whole data set. So properly mitigating this kind of emphasis may be good to the classification performance. Margin-pruning Boost mitigates the emphasis on difficult-to-classify instances in the early training phase. However, it aggravates the emphasis late because the weights of some instances filtered by the thresholding are increased in the late training phase. Compared with Gentle AdaBoost and Margin-pruning Boost, the new proposed Penalized AdaBoost mitigates the emphasis on difficult-to-classify instances from the beginning of the training. Furthermore, it highlights more competent weak classifiers which can correctly classify difficult-to-classify instances in each iteration. Thus, these difficult-to-classify instances are solved

Table 1 Comparison results of generalization errors on 26 data sets after 200 iterations.

Data sets:26	(No.S, No.F)	G-AB	M-AB	M-PB	P-AB
Australian	(690, 14)	0.1435	0.1435	0.1391	**0.1304**
Banana	(5000, 2)	0.2713	0.2804	**0.2526**	0.2809
Bankruptcy	(175, 6)	0.0400	0.0344	0.0400	**0.0286**
Banknote	(1372, 4)	**0.0022**	0.0372	0.0058	0.0080
Breast Cancer	(569, 4)	**0.0246**	0.0422	0.0669	0.0281
Blood Transfusion	(748, 4)	0.2312	0.2352	**0.2218**	0.2298
Climate model	(540, 20)	**0.0593**	0.0963	0.0704	0.0630
German Numeric	(1000, 24)	0.2580	0.2950	0.2470	**0.2450**
Glass	(214, 10)	**0.0560**	0.0701	0.0886	0.0653
Gamma Telescope	(19020,10)	0.1546	0.2297	**0.1454**	0.1548
Heart Disease	(270, 13)	0.2333	0.1704	0.2000	**0.1630**
Hepatitis	(155, 19)	0.2130	0.2515	0.2129	**0.1869**
Indian Diabetes	(768, 8)	0.2578	0.2383	0.2318	**0.2253**
Indian Liver	(579, 9)	0.3005	0.3230	**0.2850**	0.3126
Ionosphere	(351, 34)	0.0969	**0.0684**	0.0997	0.0826
Parkinsons	(195, 22)	**0.0872**	0.1692	0.0974	0.0923
Planning Relax	(182, 12)	0.4012	0.3519	0.4175	**0.3186**
Ringnorm	(7400, 20)	**0.0268**	0.0470	0.0277	0.0509
Spambase	(4601, 57)	0.0546	0.0895	**0.0528**	0.0617
SPECTFHeart	(267, 44)	0.2285	0.2472	0.2584	**0.1985**
Splice	(2991, 60)	0.0675	0.0919	**0.0612**	0.0639
Steel Plates	(1941, 27)	0.2324	0.2751	**0.2318**	0.2421
Twonorm	(7400, 20)	0.0305	0.0315	**0.0292**	0.0304
Waveform	(3304, 21)	0.0975	0.0962	0.0917	**0.0853**
Wine Quality	(6497, 11)	0.0054	0.0240	**0.0052**	0.0054
Wisconsin Prognostic	(198, 34)	0.2727	0.3030	0.3081	**0.2323**
Sum		3.8465	4.2421	3.8880	**3.5857**
Comparison to G-AB		0.0000 improved	0.3956 degraded	0.0415 degraded	0.2608 improved

Table 2 Comparison results of generalization errors on 26 databases after 500 iterations.

Data sets:26	(No.S, No.F)	G-AB	M-AB	M-PB	P-AB
Australian	(690, 14)	0.1638	0.1435	0.1377	**0.1333**
Banana	(5000, 2)	0.2725	0.2804	**0.2547**	0.2738
Bankruptcy	(175, 6)	0.0400	**0.0344**	0.0400	0.0458
Banknote	(1372, 4)	**0.0036**	0.0372	0.0058	0.0058
Breast Cancer	(569, 4)	**0.0246**	0.0369	0.0756	**0.0246**
Blood Transfusion	(748, 4)	0.2540	0.2352	0.2272	**0.2258**
Climate model	(540, 20)	0.0648	0.0963	0.0722	**0.0593**
German Numeric	(1000, 24)	0.2730	0.2950	0.2490	**0.2430**
Glass	(214, 10)	0.0607	0.0748	0.1025	**0.0606**
Gamma Telescope	(19020,10)	0.1519	0.2292	**0.1424**	0.1473
Heart Disease	(270, 13)	0.2481	**0.1704**	0.2000	0.1852
Hepatitis	(155, 19)	0.2130	0.2515	0.2129	**0.2128**
Indian Diabetes	(768, 8)	0.2526	0.2383	0.2409	**0.2266**
Indian Liver	(579, 9)	0.3074	0.3558	**0.2953**	0.3057
Ionosphere	(351, 34)	0.0940	**0.0627**	0.0997	0.0912
Parkinsons	(195, 22)	0.0974	0.1641	0.0974	**0.0872**
Planning Relax	(182, 12)	0.4397	0.3519	0.4175	**0.3462**
Ringnorm	(7400, 20)	**0.0249**	0.0309	0.0251	0.0318
Spambase	(4601, 57)	0.0574	0.0895	**0.0517**	0.0572
SPECTFHeart	(267, 44)	0.2285	0.2584	0.2772	**0.2022**
Splice	(2991, 60)	0.0746	0.0919	0.0622	**0.0605**
Steel Plates	(1941, 27)	0.2293	0.2777	**0.2267**	0.2334
Twonorm	(7400, 20)	0.0296	0.0286	0.0296	**0.0276**
Waveform	(3304, 21)	0.0984	0.0962	0.095	**0.0881**
Wine Quality	(6497, 11)	0.0054	0.0231	0.0060	**0.0051**
Wisconsin Prognostic	(198, 34)	0.2828	0.3030	0.3081	**0.2374**
Sum		3.9920	4.2569	3.9524	**3.6175**
Comparison to G-AB		0.0000 improved	0.2649 degraded	0.0396 improved	0.3745 improved
Parallel comparison to Tab.1		0.1455 degraded	0.0148 degraded	0.0644 degraded	0.0318 degraded

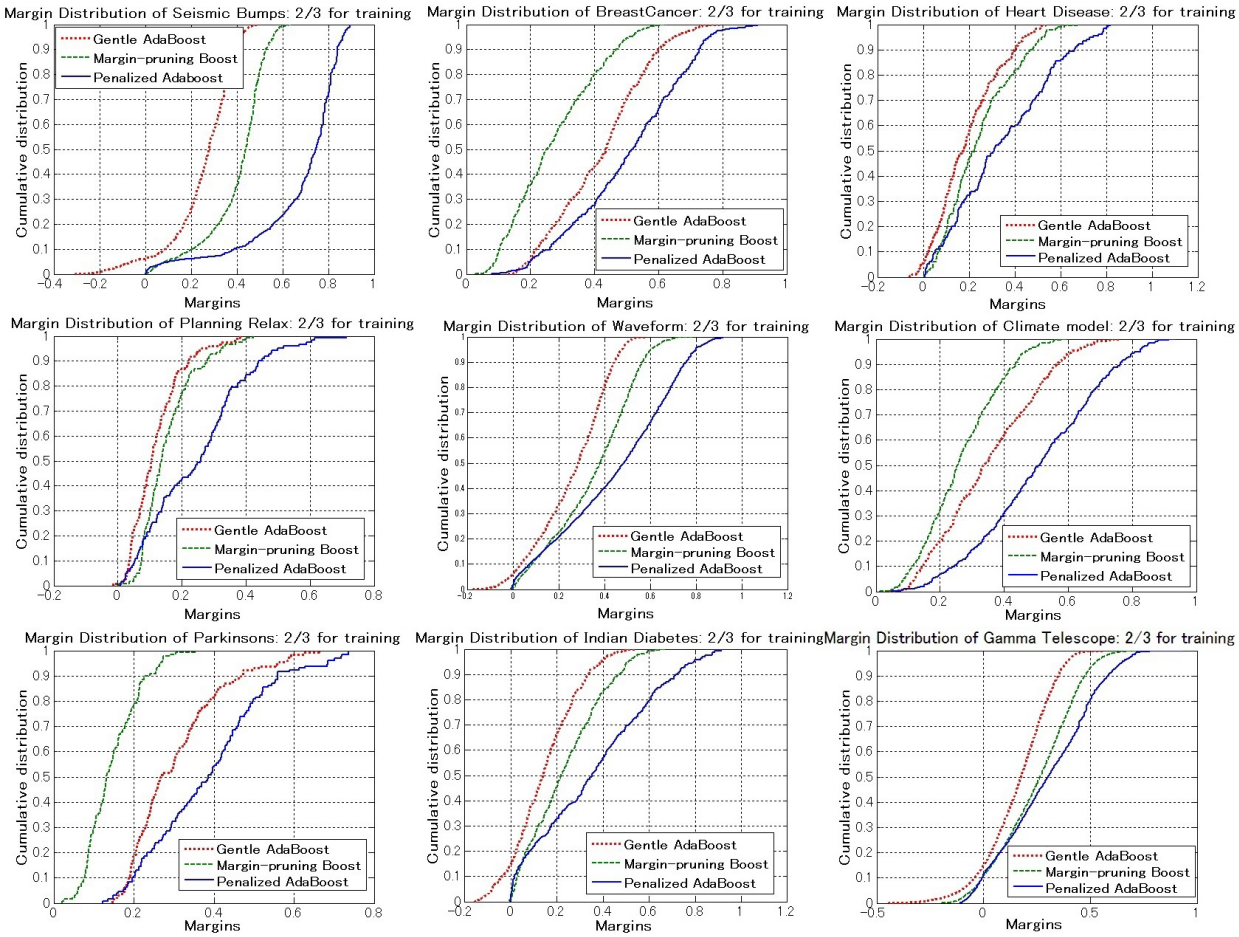


Fig. 5 Margin distributions of different data sets after 100 iterations.

gradually so that it can avoid the situation that the most difficult instances dominate the weight distribution to cause "classifier distortion" in the late training phase. Figure 5 shows some examples of margin distributions. All these margin distributions demonstrate that Penalized AdaBoost is more effective at enlarging the margins of the whole data set compared with Gentle AdaBoost and Margin-pruning Boost. We noticed that the minimal margin of Penalized AdaBoost may not be the best. However, its whole margin distribution is better than those of other variants. These margin distributions demonstrated that Penalized AdaBoost fits for the majority instances more than instances with minimal margins.

On the other hand, for the noise-like instances whose weights are large and whose margins are negative, Gentle AdaBoost classifies them as positive or negative instances. However, Penalized AdaBoost resets their summed weak hypotheses to be 0. This means Penalized AdaBoost does not predict the class of these noise-like instances because their current summed weak hypotheses can not correctly classify them. After resetting, Penalized AdaBoost continues to calculate weak hypotheses for these instances. If these instances are actually noise, their weak hypotheses will be reset to be 0 again in future runs. Therefore, their weights will keep small to avoid "classifier distortion" during the whole training. If these instances are not noise, their weak hypotheses after the resetting will be kept. Thereby, Penalized AdaBoost will predict the class of these instances. Since Penalized AdaBoost predicts the class of noise-like instances more carefully than Gentle AdaBoost, it can achieve a better generalization error than Gentle AdaBoost.

6. Conclusion

This paper introduced a novel method Penalized AdaBoost, which utilizes the margins to reduce the generalization errors. The novel contributions of this paper are as follows:

(1) It improves Margin-pruning Boost by using an adaptive resetting technique which is controlled by the current margins. The resetting technique mitigates the emphasis on the most difficult instances.

(2) It introduces a penalty policy to restrain the deterioration of small margins. Since the penalty policy corrects the wrong prediction of instances with small margins from the beginning of the boosting process, it is effective at reducing the absolute error of the strong hypothesis.

Both contributions are efficient in avoiding the "classifier distortion" since they prevent the weights of the most

Table 3 Average time of one implementation.

Gentle AdaBoost	Penalized AdaBoost
0.001687 second	0.001876 second

difficult instances from becoming too large. Moreover, unlike SoftBoost and Regularized AdaBoost, the calculation in Penalized AdaBoost is simple and the running time of each cycle is similar to that of Gentle AdaBoost. We can see that from Table 3. Here we compared our Penalized AdaBoost with Gentle AdaBoost in terms of the running time per cycle on a data set BreastCancer. The source code of Gentle and Penalized AdaBoost is programmed by utilizing the Matlab AdaBoost Toolbox mentioned in Sect. 4. The results in Table 3 were gathered from a notebook whose CPU is Intel i5 2.5GHZ (dual core) and memory is 8 GB. This paper only discussed binary classification tasks by Penalized AdaBoost. In our future work, we will extend this method to cope with multi-class problems.

References

[1] Y. Freund and R.E. Schapire, "A short introduction to boosting," Journal of Japanese Society for Artificial Intelligence, vol.14, no.5, pp.771–780, 1999.
[2] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," pp.I-511–I-518, 2001.
[3] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," The Annals of Statistics, vol.26, no.5, pp.1651–1686, 1998.
[4] R.E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," Machine Learning, vol.37, no.3, pp.297–336, 1999.
[5] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," The Annals of Statistics, vol.28, no.2, pp.337–374, 2000.
[6] E. Grossmann, "Adatree: Boosting a weak classifier into a decision tree," In IEEE Workshop on CVPR, pp.105–112, 2004.
[7] B. Paul, G. Athithan, and M.N. Murty, "Speeding up AdaBoost classifier with random projection," in 7th International Conference on Advances in Pattern Recognition, pp.251–254, 2009.
[8] C. Sun, J. Hu, and K.-M. Lam, "Feature subset selection for efficient adaboost training," ICME, pp.1–6, 2011.
[9] M. Seyedhosseini, A.R.C. Paiva, and T. Tasdizen, "Fast adaboost training using weighted novelty selection," Proc. International Conference on Neural Networks, pp.1245–1250, 2011.
[10] Y. Freund, "An adaptive version of the boost by majority algorithm," Proc. Twelfth Annual Conference on Computational Learning Theory, pp.102–113, 1999.
[11] A.J. Ferreira and M.T. Figueiredo, "Boosting algorithms: A review of methods, theory, and applications," in Ensemble Machine Learning: Methods and Applications, pp.35–85, 2012.
[12] C. Domingo and O. Watanabe, "Madaboost: a modification of adaboost," 13th Annual Conference on Computational Learning Theory, pp.180–189, 2000.
[13] R. Servedio, "Smooth boosting and learning with malicious noise," Journal of Machine Learning Research, vol.4, pp.633–648, 2003.
[14] J.K. Bradley and R.E. Schapire, "Filterboost: Regression and classification on large datasets," NIPS, pp.185–192, 2007.
[15] Y. Sun, J. Li, and W. Hager, "Two new regularized adaboost algorithms," in International Conference on Machine Learning and Applications, pp.41–48, 2004.
[16] O.T.R. Gunnar and R.M. Klaus, "Soft margins for adaboost," Machine Learning, vol.42, no.3, pp.287–320, 2001.
[17] Y. Freund, "A more robust boosting algorithm," preprint available at http://arxiv.org/abs/0905.2138, 2009.
[18] P.K. Mallapragada, R. Jin, A.K. Jain, and Y. Liu, "Semi-boost:boosting for semi-supervised learning," IEEE Trans. Pattern Anal. Mach. Intell., vol.31, no.11, pp.2000–2014, 2009.
[19] K. Chen and S. Wang, "Semi-supervised learning via regularized boosting working on multiple semi-supervised assumptions," IEEE Trans. Pattern Anal. Mach. Intell., vol.33, no.1, pp.129–143, 2011.
[20] S.Z. Li and Z. Zhang, "Floatboost learning and statistical face detection," IEEE Trans. Pattern Anal. Mach. Intell., vol.26, no.9, pp.1112–1123, 2004.
[21] A. Demiriz, K.P. Bennett, and J.S. Taylor, "Linear programming boosting via column generation," Machine Learning, vol.46, no.1-3, pp.225–254, 2002.
[22] H. Li and C. Shen, "Boosting the minimum margin: LPBoost vs. AdaBoost," Digital Image Computing: Techniques and Applications (DICTA), pp.533–539, 2008.
[23] A. Vezhnevets and V. Vezhnevets, "Modest adaboost - teaching adaboost to generalize better," Graphicon, vol.12, no.5, pp.987–997, 2005.
[24] Y.Z.J. Thongkam, G. Xu, and F. Huang, "Breast cancer survivability via adaboost algorithms," Australian Workshop on Health Data and Knowledge Management, pp.55–64, 2008.
[25] Y. Lu, Q. Tian, and T. Huang, "Interactive boosting for image classification," Proceedings of the 7th International Conference on Multiple Classifier Systems, pp.180–189, 2007.
[26] J.J. Rodríguez and J. Maudes, "Boosting recombined weak classifiers," Pattern Recognition Letters, vol.29, no.8, pp.1049–1059, 2008.
[27] M. Warmuth, K. Glocer, and G. Ratsch, "Boosting algorithms for maximizing the soft margin," Adavances in NIPS, pp.1–8, 2007.
[28] M.K. Warmuth, K.A. Glocer, and S.V.N. Vishwanathan, "Entropy regularized lpboost," Proceedings of the 19th International Conference on Algorithmic Learning Theory, pp.256–271, 2008.
[29] D. Hegazy and J. Denzler, "Performance comparison and evaluation of adaboost and softboost algorithms on generic object recognition," World Academy of Science, Engineering and Technology, pp.70–74, 2008.
[30] S. Wu and H. Nagahashi, "A new method for solving overfitting problem of gentle adaboost," SPIE 9069, Fifth International Conference on Graphic and Image Processing, pp.1–6, 2014.
[31] K. Bache and M. Lichman, "Uci machine learning repository," http://archive.ics.uci.edu/ml/, pp.Irvine, CA: University of California, School of Information and Computer Science, 2013.
[32] A. Vezhnevets and V. Vezhnevets, "Gml adaboost matlab toolbox manual," http://graphics.cs.msu.ru/ru/science/research/machinelearning/adaboosttoolbox, pp.1–12.

Shuqiong Wu received her B.S.Eng. degrees from Beihang University in 2008 and 2011 respectively. Now she is a Ph.D. candidate in Department of Computational Intelligence and Systems Science, Tokyo Institute of Technology. Her major is artificial intelligence and her research interest includes machine learning, pattern recognition, object detection and image processing.

Hiroshi Nagahashi received his B.S. and Dr.Eng. degrees from Tokyo Institute of Technology in 1975 and 1980, respectively. Since 1990, he has been with Imaging Science and Engineering Laboratory, Tokyo Institute of Technology, where he is currently a professor. His research interests include pattern recognition, computer graphics, image processing, and computer vision.