How to Make AdaBoost.M1 Work for Weak Base Classifiers by Changing Only One Line of the Code

Günther Eibl and Karl Peter Pfeiffer

Institute of Biostatistics, Innsbruck, Austria
guenther.eibl@uibk.ac.at

Abstract. If one has a multiclass classification problem and wants to boost a multiclass base classifier AdaBoost.M1 is a well known and widely applicated boosting algorithm. However AdaBoost.M1 does not work, if the base classifier is too weak. We show, that with a modification of only one line of AdaBoost.M1 one can make it usable for weak base classifiers, too. The resulting classifier AdaBoost.M1W is guaranteed to minimize an upper bound for a performance measure, called the guessing error, as long as the base classifier is better than random guessing. The usability of AdaBoost.M1W could be clearly demonstrated experimentally.

1 Introduction

A weak classifier is a map $h : X \to G$ (with $G = \{1, \ldots, |G|\}$), which assigns an object with measurements $x \in X$ to one of $|G|$ prespecified groups with a high error rate. The task of a boosting algorithm is to turn a weak classifier into a strong classifier, that has a low error rate. To simplify notation we define, that $\arg\max_{g \in G} u(g)$ is the group g, which maximizes the function u.

Most papers about boosting theory consider twoclass classification problems ($|G|$=2). Multiclass problems can then be reduced to twoclass problems using for example error-correcting codes [1, 2, 4, 5].

However if one has a multiclass problem and also a base classifier for multiclass problems like decision trees one would prefer a more direct boosting method.

Freund and Schapire [3] proposed the algorithm AdaBoost.M1 (Fig.1), which is a straightforward generalization of AdaBoost for 2 groups for the multiclass problem using multiclass base classifiers. One of the main ideas of the algorithm is to maintain a distribution D of weights over the learning set $\mathcal{L} = \{(x_1, g_1), \ldots, (x_N, g_N);\ x_i \in X,\ g_i \in G\}$. The weight of this distribution on training example i on round t is denoted by $D_t(i)$. On each round the weights of incorrectly classified examples are increased so that the weak learner h is forced to focus on the "hard" examples in the training set. The goal of the weak learner is to find a hypothesis h_t appropriate for the distribution D_t. The goodness of

h_t is measured by its weighted error rate

$$\epsilon_t = \sum_i D_t(i) I(h_t(x_i) \neq g_i)$$

with I denoting the indicator function. In practice a subset of the training examples is sampled according to D_t, and these (unweighted) resampled examples are used to train the weak learner. After h_t has been received, AdaBoost.M1 chooses α_t, which measures the importance assigned to h_t. The sampling distribution D_t is next updated, where the weights of examples misclassified by h_t are increased and the weights of correctly classified examples are decreased. Thus, the weight tends to concentrate on the "hard" examples. The final hypothesis is a weighted majority vote of T weak hypotheses where α_t is the weight assigned to h_t.

The most important property of AdaBoost.M1 concerns its ability to reduce the training error. An exponential decrease of an upper bound of the training error rate is guaranteed as long as the error rates of the base classifiers are less than 1/2. This also leads to the criterion to stop, if ϵ_t is greater or equal to 1/2.

However for more than two groups the condition, that the error rate of the base classifier is less than 1/2, can be too restrictive, if one uses weak base classifiers as for example decision stumps.

Freund and Schapire [3] overcame this problem with the introduction of the pseudo-loss of a confidence-rated classifier $h : X \times G \to [0, 1]$

$$\text{pseudo} - \text{loss}(h) = \frac{1}{2}\left(1 - h(x_i, g_i) + \sum_{g \neq g_i} \frac{1}{G-1} h(x_i, g)\right) \quad .$$

In the algorithm AdaBoost.M2 each base classifier has to minimize the pseudo-loss instead of the error rate. As long as the pseudo-loss is less than 1/2, which is a very weak condition, an exponential decrease of an upper bound of the training error rate is guaranteed. The drawback of this approach lies in the need to redesign the base classifier in order to give confidences and to minimize the pseudo-loss.

In this paper we make a simple modification in AdaBoost.M1 in order to make it applicable for weak base classifiers. We call this algorithm AdaBoost.M1W. The modification consists of a change in only one line of the code, which considers the definition of $\alpha_t(\epsilon_t)$. First we give an ad-hoc derivation of the algorithm. Then we derive a theorem, which states, that an upper bound for a performance measure, called the guessing error, is guaranteed to be minimized as long as the base classifier is better than random guessing. Finally we performed experiments with multiclass datasets to look, if the algorithm also works in practice. We can clearly show, that this is the case, because for all 8 datasets, where the base classifier is too weak for AdaBoost.M1, AdaBoost.M1W still works.

Input: learning set $\mathcal{L} = \{(x_1, g_1), \ldots, (x_N, g_N);\ x_i \in X,\ g_i \in G\}$,
$G = \{1, \ldots, |G|\}$, classifier of the form $h : X \to G$.
T: number of boosting rounds

Initialization: $D_1(i) = \frac{1}{N}$.

For $t = 1, \ldots, T$**:**

- Train the weak classifier h_t with distribution D_t, where h_t should minimize the weighted error rate
$$\epsilon_t = \sum_i D_t(i) I(h_t(x_i) \neq g_i) \ .$$
- If $\epsilon_t \geq 1/2$: goto output with $T := t - 1$.
- Set
$$\alpha_t = \ln\left(\frac{1 - \epsilon_t}{\epsilon}\right) \ .$$
- Update D:
$$D_{t+1}(i) = D_t(i) e^{-\alpha_t I(h_t(x_i) = g_i)} / Z_t$$
where Z_t is a normalization factor (chosen so that D_{t+1} is a distribution)

Output: Set the final classifier $H(x)$:

$$H(x) = \arg\max_{g \in G} f(x, g) = \arg\max_{g \in G} \left(\sum_{t=1}^{T} \alpha_t I(h_t(x) = g)\right) \ .$$

Fig. 1. Algorithm AdaBoost.M1.

2 Ad-hoc Derivation of AdaBoost.M1W

When we analyzed AdaBoost.M1 we wondered, why AdaBoost.M1 does not work with weak base classifiers and if it is possible to modify AdaBoost.M1 to make it work with weak base classifiers.

We start by looking at the combination step

$$H(x) = \arg\max_{g \in G} \left(\sum_{t=1}^{T} \alpha_t I(h_t(x) = g)\right) \ .$$

There each base classifier h_t gives a vote for the group $h_t(x)$, to which it would assign x. The votings are weighted by the factor α_t, which is bigger, if the base classifier is better. The key point for the modification is the property of the algorithm AdaBoost.M1, that

$$\alpha_t = \ln\left(\frac{1 - \epsilon_t}{\epsilon}\right) \geq 0 \Leftrightarrow \epsilon_t \leq \frac{1}{2} \ . \tag{1}$$

If the error rate is bigger than $1/2$, the weight α_t gets negative, so the ensemble classifier H does the opposite of what the base classifier h_t proposes. In

AdaBoost.M1 α_t does not get negative, because in the derivation of the bound for the training error of AdaBoost.M1 α_t is assumed to be positive and AdaBoost.M1 stops, if $\epsilon_t \geq \frac{1}{2}$ (and therefore $\alpha_t \leq 0$ because of (1)). Nevertheless this was the starting point for our modification.

If one has a base classifier h with error rate greater than $1/2$ but better than random guessing (which has an expected error rate of $1 - 1/|G|$), the ensemble classifier H should not do the opposite of what the base classifier h proposes. So we wanted to find a choice for $\alpha_t(\epsilon_t)$ such that

$$\alpha_t \geq 0 \Leftrightarrow \epsilon_t \leq 1 - \frac{1}{|G|} \ . \tag{2}$$

To derive $\alpha_t(\epsilon_t)$ we assumed $\alpha_t(\epsilon_t)$ to be basically of the same form as α_t in AdaBoost.M1, so we set

$$\alpha_t(\epsilon_t) = \ln\left(\frac{a_n\epsilon_t + b_n}{a_d\epsilon_t + b_d}\right) =: \ln\left(z(\epsilon_t)\right) \tag{3}$$

where n and d are subscripts for the nominator and denominator respectively. Then we wanted α_t to fulfill (2) and two additional conditions, which are also fulfilled by $\alpha_t(\epsilon_t)$ of AdaBoost.M1:

$$\begin{aligned} \alpha_t(1 - \frac{1}{|G|}) &= 0 \\ \alpha_t \to -\infty \ \text{ for } \ \epsilon_t &\to 1 \\ \alpha_t \to \infty \ \text{ for } \ \epsilon_t &\to 0 \ . \end{aligned}$$

For $z(\epsilon_t)$ this means that

$$\begin{aligned} z(1 - \frac{1}{|G|}) &= 1 \\ z(1) &= 0 \\ \epsilon_t = 0 \Rightarrow \ &\text{Denominator of } z = 0 \ . \end{aligned}$$

These conditions directly results in the following conditions for the 4 constants

$$\begin{aligned} a_n\left(1 - \frac{1}{|G|}\right) + b_n &= a_d\left(1 - \frac{1}{|G|}\right) + b_d \\ b_n &= -a_n \\ b_d &= 0 \ . \end{aligned}$$

Substitution for b_n and b_d in the first equation and solving for a_d leads to

$$a_n = a_d(1 - |G|) \ .$$

Now we substitute the constants $a_n,\ b_n$ and b_d in (3), a_d gets cancelled, and we get

$$\alpha_t = \ln\left(\frac{(|G| - 1)(1 - \epsilon_t)}{\epsilon_t}\right). \tag{4}$$

Note, that up to this point this is just an ad-hoc modification without any proof for a decrease in the error rate. So we also don't have a stopping criterion any more. An intuitive ad-hoc stopping criterion would stop, if

$$\epsilon_t \geq 1 - \frac{1}{|G|} \quad .$$

For the experiments we stopped after a big, prespecified number T of boosting rounds and investigated, if the stopping criterion above would have done well. Since the rest of the algorithm AdaBoost.M1 is left untouched we can already write down the algorithm in Fig. 2.

Input: learning set $\mathcal{L} = \{(x_1, g_1), \ldots, (x_N, g_N);\ x_i \in X,\ g_i \in G\}$,
$G = \{1, \ldots, |G|\}$, classifier of the form $h : X \to G$.
T: number of boosting rounds

Initialization: $D_1(i) = \frac{1}{N}$.

For $t = 1, \ldots, T$**:**

- Train the weak classifier h_t with distribution D_t, where h_t should minimize the weighted error rate
$$\epsilon_t = \sum_i D_t(i) I(h_t(x_i) \neq g_i) \quad .$$
- Set
$$\alpha_t = \ln\left(\frac{(|G|-1)(1-\epsilon_t)}{\epsilon_t}\right) \quad .$$
- Update D:
$$D_{t+1}(i) = D_t(i) e^{-\alpha_t I(h_t(x_i)=g_i)} / Z_t$$
where Z_t is a normalization factor (chosen so that D_{t+1} is a distribution)

Output: Set the final classifier $H(x)$:

$$H(x) = \arg\max_{g \in G} f(x, g) = \arg\max_{g \in G} \left(\sum_{t=1}^{T} \alpha_t I(h_t(x) = g)\right) \quad .$$

Fig. 2. Algorithm AdaBoost.M1W.

3 Theoretical Analysis of AdaBoost.M1W

Due to suggestions of the reviewers we also made a theoretical analysis of AdaBoost.M1W. We can show, that the algorithm doesn't minimize an upper bound for the training error, but an upper bound for a new performance measure, which we call the guessing error. This performance measure compares the final classifier

with random guessing, which has a training error rate of $1 - 1/|G|$. The guessing error *guesserr* is defined as the proportion of examples, where the classifier performs worse than random guessing.

Definition 1. *A classifier $f : X \times G \to [0, 1]$ makes a guessing error in classifying an object x coming from group g, if*

$$f(x, g) < \frac{1}{|G|} \ .$$

The corresponding estimate of the expected guessing error using the training set is called guesserr:

$$guesserr := \sum_{i=1}^{N} I\left(f(x_i, g_i) < \frac{1}{|G|}\right) \ .$$

Note, that by dividing f from AdaBoost.M1W by $\sum_t \alpha_t$ we ensure, that $f(x, g) \in [0, 1]$.

The following theorem guarantees an exponential decrease of the guessing error of AdaBoost.M1W as long as the base classifier is better than random guessing.

Theorem 1. *If all base classifiers $h_t : X \to G$ satisfy*

$$\epsilon_t = \sum_i D_t(i) I(h_t(x_i) \neq g_i) \leq 1 - 1/|G| - \delta$$

for $\delta \in (0, 1-1/|G|)$, then the guessing error of the training set for AdaBoost.M1W fulfills

$$guesserr < \prod_t \frac{\epsilon_t^{1-1/|G|} \ (1 - \epsilon_t)^{1/|G|}}{(1 - 1/|G|)^{1-1/|G|} \ (1/|G|)^{1/|G|}} \leq e^{-\delta^2 T} \ .$$

Proof. (i) Similar to the calculations used to bound the error rate of AdaBoost we begin by bounding *guesserr* in terms of the normalization constants Z_t: We make a guessing error for example i, if

$$\frac{f(x_i, g_i)}{\sum_t \alpha_t} < \frac{1}{|G|} \Rightarrow e^{-(f(x_i, g_i) - \sum_t \alpha_t / |G|)} > 1 \ .$$

So

$$guesserr := \sum_{i=1}^{N} I\left(\frac{f(x_i, g_i)}{\sum_t \alpha_t} < \frac{1}{|G|}\right) < \sum_{i=1}^{N} e^{-(f(x_i, g_i) - \sum_t \alpha_t / |G|)} \ . \qquad (5)$$

Now we unravel the update rule

$$1 = \sum_i D_{t+1}(i) = \sum_i D_t(i) \frac{e^{-\alpha_t I(h_t(x_i) = g_i)}}{Z_t} = \dots$$

$$= \frac{1}{\prod_s Z_s} \frac{1}{N} \sum_i \prod_{s=1}^{t} e^{-\alpha_s I(h_s(x_i) = g_i)} = \frac{1}{\prod_s Z_s} \frac{1}{N} \sum_i e^{-f(x_i, g_i)} \ .$$

So we get

$$\prod_t Z_t = \frac{1}{N} \sum_i e^{-f(x_i, g_i)}$$

and, together with (5), we get

$$guesserr \leq \left(\prod_t Z_t \right) \left(e^{\sum_t \alpha_t / |G|} \right) = \prod_t e^{\alpha_t / |G|} Z_t \ . \tag{6}$$

(ii) Now we bound $\prod_t e^{\alpha_t/|G|} Z_t$:

$$\begin{aligned}
\prod_t e^{\alpha_t/|G|} Z_t &= \prod_t \left(e^{\alpha_t/|G|} \sum_i D_t(i) e^{-\alpha_t I(h_t(x_i)=g_i)} \right) . \\
&= \prod_t e^{\alpha_t/|G|} \left(\sum_{i; h_t(x_i)=g_i} D_t(i) e^{-\alpha_t} + \sum_{i; h_t(x_i) \neq g_i} D_t(i) \right) \\
&= \prod_t e^{\alpha_t/|G|} \left(e^{-\alpha_t}(1-\epsilon_t) + \epsilon_t \right) \\
&= \prod_t \left(\frac{(|G|-1)(1-\epsilon_t)}{\epsilon_t} \right)^{1/|G|} \left(\frac{\epsilon_t}{|G|-1} + \epsilon_t \right) \\
&= \prod_t \frac{\epsilon_t^{1-1/|G|} \ (1-\epsilon_t)^{1/|G|}}{(1-1/|G|)^{1-1/|G|} \ (1/|G|)^{1/|G|}} \ .
\end{aligned}$$

So together with (6) we get

$$guesserr \leq \prod_t \frac{\epsilon_t^{1-1/|G|} \ (1-\epsilon_t)^{1/|G|}}{(1-1/|G|)^{1-1/|G|} \ (1/|G|)^{1/|G|}} \ . \tag{7}$$

(iii) Now we show, that this bound for *guesserr* decreases exponentially, if $\epsilon_t = 1 - 1/|G| - \delta$ with $\delta \in (0, 1 - 1/|G|)$ for all t. We can rewrite (7) as

$$guesserr \leq \prod_t \left(1 - \frac{\delta}{1-1/|G|} \right)^{1-1/|G|} \left(1 + \frac{\delta}{1/|G|} \right)^{1/|G|}$$

and bound both terms using the binomial series. The series of the first term have only negative terms. We stop after the term of first order and get

$$\left(1 - \frac{\delta}{1-1/|G|} \right)^{1-1/|G|} \leq 1 - \delta \ .$$

The series of the second term have both positive and negative terms. We stop after the positive term of first order and get

$$\left(1 + \frac{\delta}{1/|G|} \right)^{1/|G|} \leq 1 + \delta \ .$$

Thus

$$guesserr \leq \prod_t (1-\delta)(1+\delta) = \prod_t (1-\delta^2) \ .$$

Using $1 + x \leq e^x$ for $x \leq 0$ leads to

$$guesserr \leq e^{-\delta^2 T} \ . \tag{8}$$

□

Due to the theorem not only the algorithm but also the ad-hoc stopping criterion of the previous section is theoretically confirmed now.

There are some generalization possibilities of AdaBoost.M1W: the definition of the guessing error and the theorem can be generalized for any $C \in (0, 1/2]$ replacing $1/|G|$ in a straightforward way leading to the performance measure

$$err_C := \sum_{i=1}^{N} I\left(f(x_i, g_i) < C\right)$$

and

$$\alpha_t = \ln\left(\frac{(1-C)(1-\epsilon_t)}{C\epsilon_t}\right) \ .$$

This generalization also contains AdaBoost.M1 by setting $C = 1/2$. One can easily verify, that for this case the theorem above and the theorem given in [3] coincide.

Another apparent generalization would regard confidence-rated base classifiers $h : X \times G \to [0, 1]$ instead of base classifiers $h : X \to G$. We are currently working on generalizing the algorithm and the theorem to this case and are very confident to finish this work soon.

4 Experiments

In our experiments we analyzed 9 multiclass datasets (Table 1) with both the algorithm AdaBoost.M1 and AdaBoost.M1W using decision stumps as base classifiers. The aim is to compare AdaBoost.M1 with AdaBoost.M1W. We decided not to compare it with AdaBoost.M2, because the latter uses confidence-rated base classifiers, which could give it a spurious advantage especially for big datasets [6]. However we plan to compare the generalization of AdaBoost.M1W, which also uses confidence-rated base classifiers, to AdaBoost.M2.

The main question to be answered by the experiments is, if AdaBoost.M1W is able to boost base classifiers with error rates greater than 1/2. The answer to this question is yes. For the 8 datasets, where the error rate of a single decision stump exceeds 1/2, AdaBoost.M1 failed, because for all 8 datasets it couldn't decrease the training error rate at all, whereas AdaBoost.M1W worked for all 8 datasets (Table 1 and Fig.3).

Since AdaBoost.M1 didn't work for any of these 8 datasets we wanted to make an additional check, that the algorithms are programmed properly. The

Table 1. Properties of the databases, initial and minimal training error of AdaBoost.M1W, $H_t := \sum_{s=1}^{t} \alpha_s h_s$.

database	N	♯ groups	♯ variables	$err(h_1)$	$\arg\min_t err(H_t)$
digitbreiman	5000	10	7	81.1%	25.6%
letter	20000	26	16	92.4%	53.0%
optdigits	5620	10	64	79.7%	0.0%
pendigits	10992	10	16	79.3%	21.8%
satimage	6435	6	34	55.3%	20.7%
segmentation	2310	7	19	71.1%	6.8%
vehicle	846	4	18	58.1%	32.6%
vowel	990	11	10	82.8%	49.8%
waveform	5000	3	21	42.7%	15.1%

waveform dataset is the only one, where the error rate of a single decision stump is less than 1/2 and therefore AdaBoost.M1 (which was programmed without stopping criterion) is expected to work. This is the case, both algorithms can decrease the training error from 42.7% below 20 % (Fig. 4)(the Bayes error for this dataset is about 14%).

It was surprising, that AdaBoost.M1W was better than AdaBoost.M1 for this dataset. The base classifiers of AdaBoost.M1 had error rates greater than 1/2 already at iteration 35, the error rates of the base classifiers of AdaBoost.M1W were greater than $1 - 1/|G|$ from iteration 165 on. So AdaBoost.M1W is an ensemble of weaker trees, but the ensemble is bigger than the one of AdaBoost.M1. We don't want to overrate the result, that AdaBoost.M1W also outperformed AdaBoost.M1, when the weak classifier had an initial error rate below 1/2, because it is a result for just for one dataset. Further experiments with other datasets and other base classifiers are necessary to confirm this result.

We also investigated the stopping criterion, which would stop the algorithm at the first round t_{stop}, where $\epsilon_t \geq 1 - 1/|G|$. Figure 3 shows, that the stopping criterion is reasonable, but often stops before the training error has reached its minimum. This fact can be explained by Fig. 5. The training errors of the base classifiers by definition reach $1 - 1/|G|$ the first time at t_{stop}, but then they can get below $1 - 1/|G|$ again. When the training errors of the base classifiers are consistently above $1 - 1/|G|$ (right of the second vertical line) the training error of the ensemble isn't improved any more. So the stopping criterion makes sense, but should be treated in a softer way. For example one could stop, if the last 5 training errors of the base classifiers are all above $1 - 1/|G|$.

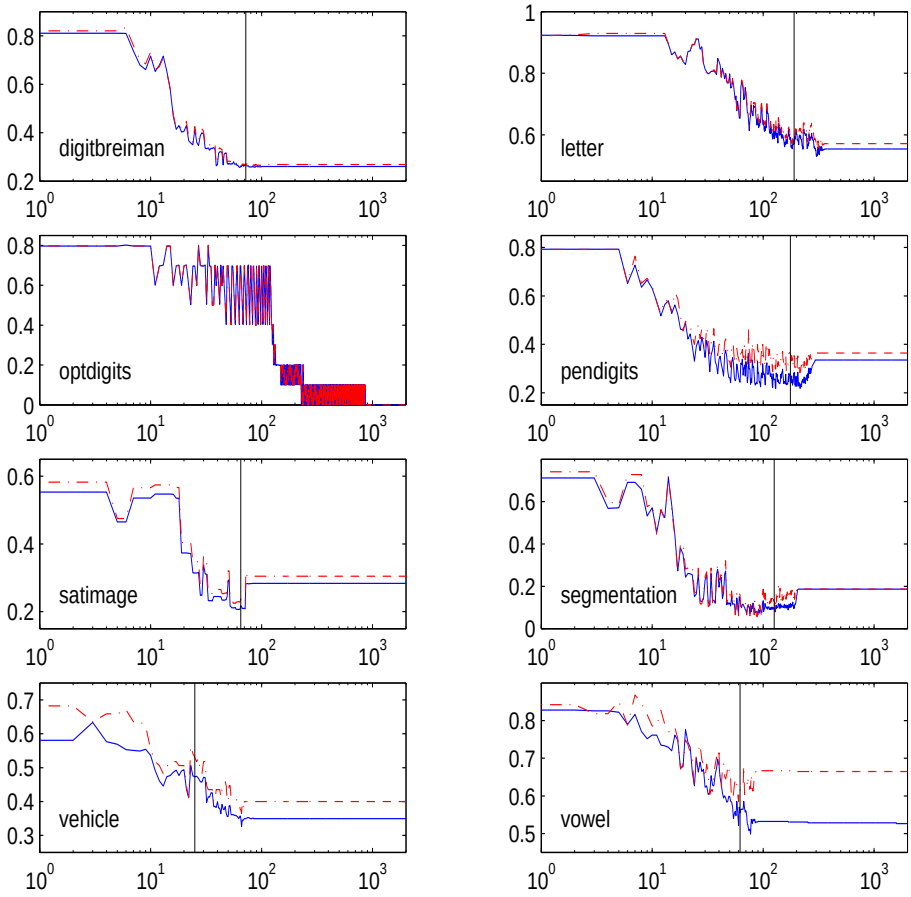

Fig. 3. Training (solid) and test error (dash-dotted) of AdaBoost.M1W dependent on the number of boosting rounds. The vertical line denotes t_{stop}.

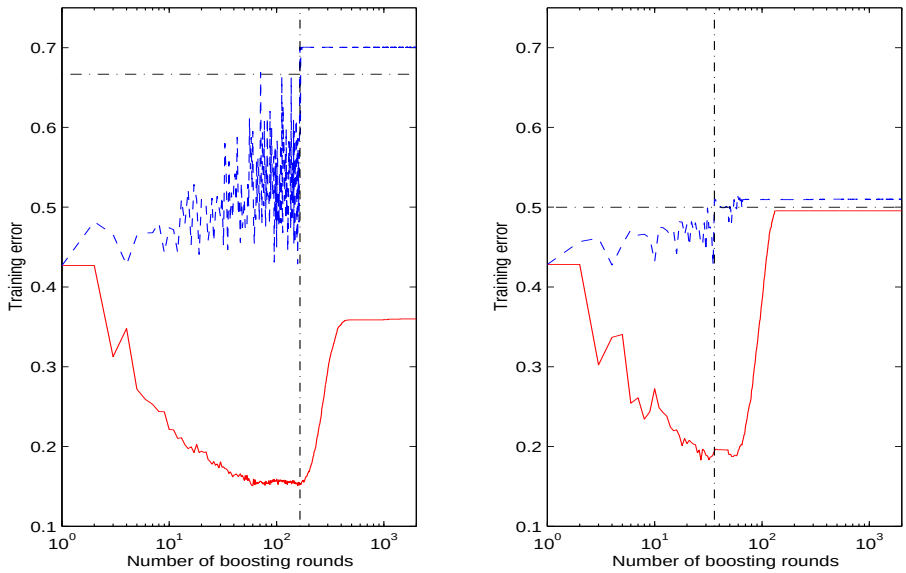

Fig. 4. Training error of the base (dashed) and the ensemble classifier (solid) for the waveform dataset for AdaBoost.M1W (left panel) and AdaBoost.M1 (right panel). The vertical line denotes t_{stop}.

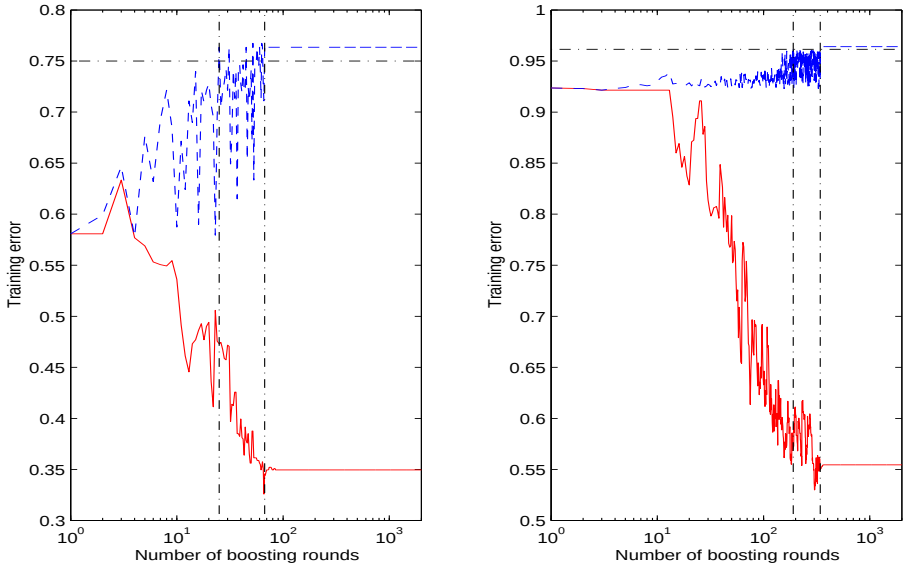

Fig. 5. Training error of the base (dashed) and the ensemble classifier (solid) for the vehicle (left panel) and letter (right panel) dataset. The first vertical line denotes t_{stop}.

5 Conclusion and Future Work

In this paper we proposed a new boosting algorithm AdaBoost.M1W, which directly boosts multiclass base classifiers for multiclass problems. The algorithm comes from the well known algorithm AdaBoost.M1. The difference to AdaBoost.M1 considers the definition of the weights of the base classifiers, which results in a change of only one line of the programming code. So everybody, who has implemented AdaBoost.M1, can easily get AdaBoost.M1W.

We introduced a performance measure, called the guessing error, which is the proportion of examples, where the final classifier is worse than random guessing. Then we derived an upper bound for this guessing error, which gets minimized exponentially fast by AdaBoost.M1W as long as the base classifiers are better than random guessing. A generalization, which contains both AdaBoost.M1W and AdaBoost.M1 and which leads to the already known upper bounds for the corresponding performance measures is straightforward.

The change of this one line has much impact, because it makes the algorithm work for weak base classifiers, which could be clearly demonstrated with experiments. AdaBoost.M1W also had a slightly better result for the one dataset, where the base classifier is strong enough for AdaBoost.M1 to work.

To explore this further we plan to make more experiments with AdaBoost.M1W for stronger base classifiers. We will also work on generalizing the algorithm for confidence-rated base classifiers.

References

1. E.L.Allwein, R.E.Schapire, Y.Singer 2000. Reducing multiclass to binary: a unifying approach for margin classifiers. Machine Learning 1, 113-141.
2. T.G.Dietterrich, G.Bakiri, 1995. Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research 2, 263-286.
3. Y.Freund, R.E.Schapire, 1997. A decision-theoretic generalization of online-learning and an application to boosting. Journal of Computer and System Sciences 55 (1), 119-139.
4. V.Guruswami, A.Sahai, 1999. Multiclass learning, boosting, and error-correcting codes. Proceedings of the Twelfth Annual Conference on Computational Learning Theory 145-155.
5. R.E.Schapire, 1997. Using output codes to boost multiclass learning problems. Machine Learning: Proceedings of the Fourteenth International Conference, 313-321.
6. R.E. Schapire, Y.Singer, 1999. Improved boosting algorithms using confidence-rated predictions. Machine Learning 37, 297-336.