Published in: T. Windeatt and F. Roli (Eds.),
Proc. 4th Int. Workshop on Multiple Classifier Systems
Springer-Verlag, LNCS, **2709**, 2003, 25–34

Error Bounds for Aggressive and Conservative AdaBoost

Ludmila I. Kuncheva
(**Please see footnote 1**)

School of Informatics, University of Wales, Bangor
Bangor, Gwynedd, LL57 1UT, United Kingdom
`l.i.kuncheva@bangor.ac.uk`

Abstract. Three AdaBoost variants are distinguished based on the strategies applied to update the weights for each new ensemble member. The classic AdaBoost due to Freund and Schapire only decreases the weights of the correctly classified objects and is conservative in this sense. All the weights are then updated through a normalization step. Other AdaBoost variants in the literature update all the weights before renormalizing (aggressive variant). Alternatively we may increase only the weights of misclassified objects and then renormalize (the second conservative variant). The three variants have different bounds on their training errors. This could indicate different generalization performances. The bounds are derived here following the proof by Freund and Schapire for the classical AdaBoost for multiple classes (AdaBoost.M1), and compared against each other. The aggressive variant and the less popular of the two conservative variants have lower error bounds than the classical AdaBoost. Also, whereas the coefficients β_i in the classical AdaBoost are found as the unique solution of a minimization problem on the bound, the aggressive and the second conservative variants have monotone increasing functions of β_i ($0 \leq \beta_i \leq 1$) as their bounds, giving infinitely many choices of β_i.[1]

1 Introduction

AdaBoost is an algorithm for designing classifier ensembles based on maintaining and manipulating a distribution of weights on the training examples. These weights are updated at each iteration to form a new training sample on which a new ensemble member is constructed.

[1] After the publication the author discovered a fault the proofs of the bounds for the Aggressive and the Conservative2 versions: The Lemma cannot be applied because the condition $r \in [0, 1]$ is not met. If the proof is correctly done, the bounds for all three versions are the same: equal to the Conservative 1 bound derived here directly for c classes. The optimal β_k for Conservative 2 is the same as for Conservative 1 and for the Aggressive version $\beta_k = \sqrt{\frac{\epsilon_k}{1-\epsilon_k}}$ leads to the same error bound on the training data: $\epsilon < 2^L \prod \sqrt{\epsilon_k(1-\epsilon_k)}$.

Looking at the variety of implementations and interpretations, it seems that AdaBoost is rather a concept than a single algorithm. For example, a subject of debate has been the resampling versus reweighing when the distribution of weights has to be applied in order to derive the next ensemble member. Not only has the question not been resolved yet but sometimes it is impossible to tell from the text of a study which of the two methods has been implemented.

In this paper we are concerned with another "technicality": the way of updating the coefficients at each step. This detail is not always mentioned in the AdaBoost studies although as we show later, it makes a difference at least to the theoretical bounds of the algorithm.

There are three general teaching strategies shown in Table 1, which we, humans, experience since an early age. In the case of a successful outcome of an experiment, we can either be rewarded or no action be taken. In the case of an unsuccessful outcome, the possibilities are no-action or punishment. Obviously, if no action is taken in both cases, nothing will be learned from the experience. Thus there are three possible combinations which we associate with the three AdaBoost variants.

Table 1. Three teaching strategies and the respective change in the weights w before the renormalization step.

Strategy	Name	w – correct	w – wrong
Reward - Punishment	Aggressive	Smaller	Larger
Reward - No-action	Conservative.1	Smaller	The same
No-action - Punishment	Conservative.2	The same	Larger

Error bounds on the training error have been derived in [3] for the Conservative.1 version. Following this proof, here we prove error bounds on the training error for the Aggressive and Conservaitve.2 version.

The rest of the paper is organized as follows. A general AdaBoost algorithm and the three variants are presented in Section 2. Section 3 contains the derivation of the bounds on the training error. A comparison is given in Section 4.

2 AdaBoost variants

The generic algorithm of AdaBoost is shown in Figure 1.

Many versions of the above algorithm live under the same name in the literature on machine learning and pattern recognition. We distinguish between the following three (see Table 1)

```
ADABOOST
Training phase
1. Given is a data set $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$.
Initialize the parameters
   · $\mathbf{w}^1 = [w_1, \ldots, w_N]$, the weights, $w_j^1 \in [0, 1]$, $\sum_{j=1}^{N} w_j^1 = 1$.
   $\left(\text{Usually } w_j^1 = \frac{1}{N}\right)$.
  · $\mathcal{D} = \emptyset$, the ensemble of classifiers.
  · $L$, the number of classifiers to train.
2. For $k = 1, \ldots, L$
   · Take a sample $S_k$ from $\mathbf{Z}$ using distribution $\mathbf{w}^k$.
   · Build a classifier $D_k$ using $S_k$ as the training set.
   · Calculate the weighted error of $D_k$ by
         $$\epsilon_k = \sum_{j=1}^{N} w_j^k l_k^j, \qquad (1)$$
   $\left(l_k^j = 1 \text{ if } D_k \text{ misclassifies } \mathbf{z}_j \text{ and } l_k^j = 0, \text{ otherwise.}\right)$.
   · If $\epsilon_k = 0$ or $\epsilon_k \geq 0.5$, the weights $w_j^k$ are reinitialized to $\frac{1}{N}$.
  · Calculate
         $$\beta_k = \frac{\epsilon_k}{1 - \epsilon_k}, \quad \text{where} \quad \epsilon_k \in (0, 0.5), \qquad (2)$$
  · Update the individual weights
         $$w_j^{k+1} = \frac{w_j^k \beta_k^{\xi(l_k^j)}}{\sum_{i=1}^{N} w_i^k \beta_k^{\xi(l_k^i)}}, \quad j = 1, \ldots, N. \qquad (3)$$
   where $\xi(l_k^j)$ is a function which specifies which of the Boosting variants we use.
3. Return $\mathcal{D}$ and $\beta_1, \ldots, \beta_L$.
Classification phase
4. Calculate the support for class $\omega_t$ by
         $$\mu_t(\mathbf{x}) = \sum_{D_k(\mathbf{x}) = \omega_t} \ln\left(\frac{1}{\beta_k}\right). \qquad (4)$$
5. The class with the maximal support is chosen as the label for $\mathbf{x}$.
```

Fig. 1. A generic description of the Boosting algorithm for classifier ensemble design

$$\begin{array}{lrcl}\text{Aggressive AdaBoost [2,4,5]} & \xi(l_k^j) & = & 1-2l_k^j; \\ \text{Conservative.1 AdaBoost [1,3]} & \xi(l_k^j) & = & 1-l_k^j; \\ \text{Conservative.2 AdaBoost} & \xi(l_k^j) & = & -l_k^j. \end{array} \tag{5}$$

The algorithm in Figure 1 differs slightly from AdaBoost.M1 [3] in that we do not perform the normalization of the weights as a separate step. This is reflected in the proofs in the next section.

3 Upper bounds of Aggressive, Conservative.1 and Conservative.2 AdaBoost

Freund and Schapire prove an upper bound on the training error of AdaBoost [3] first for the case of two classes and (Conservative.1, $\xi(l_k^j) = 1-l_k^j$). We will prove the bound for c classes straight away and derive from it the bounds for c classes for the Aggressive version and the Conservative.2 version.

The following Lemma is needed within the proof.

Lemma. *Let $a \geq 0$ and $r \in [0,1]$. Then*

$$a^r \leq 1-(1-a)r. \tag{6}$$

Proof. Take a^r to be a function of r for a fixed $a \geq 0$. The second derivative

$$\frac{\partial^2 (a^r)}{\partial r^2} = a^r (\ln a)^2, \tag{7}$$

is always nonnegative, therefore a^r is a convex function. The righthand side of inequality (6) represents a point on the line segment through points $(0,1)$ and $(1,a)$ on the curve a^r, therefore (6) holds for any $r \in [0,1]$. ■

Theorem 1. (Conservative.1) $\boxed{\xi(l_k^j) = 1-l_k^j}$ *Let ϵ be the ensemble training error and let ϵ_i, $i = 1,\ldots,L$ be the weighted training errors of the classifiers in $\mathcal{D}$, as in (1). Then*

$$\epsilon \;<\; 2^L \prod_{i=1}^{L} \sqrt{\epsilon_i(1-\epsilon_i)}. \tag{8}$$

Proof. After the initialization, the weights are updated to

$$w_j^2 = \frac{w_j^1 \beta_1^{(1-l_j^1)}}{\sum_{k=1}^N w_k^1 \beta_1^{(1-l_k^1)}} \tag{9}$$

Denote the normalizing coefficient at step i by

$$C_i = \sum_{k=1}^{N} w_k^i \beta_i^{(1-l_k^i)} \tag{10}$$

The general formula for the weights is

$$w_j^{t+1} = w_j^1 \prod_{i=1}^{t} \frac{\beta_i^{(1-l_j^i)}}{C_i}. \tag{11}$$

Denote by $\mathbf{Z}^{(-)}$ the subset of elements of the training set $\mathbf{Z}$ which are misclassified by the ensemble. The ensemble error, weighted by the initial data weights w_j^1 is

$$\epsilon = \sum_{\mathbf{z}_j \in \mathbf{Z}^{(-)}} w_j^1. \tag{12}$$

(If we assign equal initial weights of $\frac{1}{N}$ to the objects, ϵ is the proportion of misclassifications on $\mathbf{Z}$ made by the ensemble.)

Since at each step, the sum of the weights in our algorithm equals one,

$$1 \;=\; \sum_{j=1}^{N} w_j^{L+1} \;\geq\; \sum_{\mathbf{z}_j \in \mathbf{Z}^{(-)}} w_j^{L+1} \;=\; \sum_{\mathbf{z}_j \in \mathbf{Z}^{(-)}} w_j^1 \prod_{i=1}^{L} \frac{\beta_i^{(1-l_j^i)}}{C_i}. \tag{13}$$

For the ensemble to commit an error in labeling of some $\mathbf{z}_j$, the sum of weighted votes for the wrong class label in (4) must be higher than any other score, including that of the right class label. Let us split the set of L classifiers into three subsets according to their outputs for a particular $\mathbf{z}_j \in \mathbf{Z}$

$\mathcal{D}^w \subset \mathcal{D}$, the set of classifiers whose output is the winning (wrong) label;
$\mathcal{D}^+ \subset \mathcal{D}$, the set of classifiers whose output is the true label;
$\mathcal{D}^- \subset \mathcal{D}$, the set of classifiers whose output is another (wrong) label.

The support for the winning class is

$$\sum_{D_i \in \mathcal{D}^w} \ln\left(\frac{1}{\beta_i}\right) \;\geq\; \sum_{D_i \in \mathcal{D}^+} \ln\left(\frac{1}{\beta_i}\right) \tag{14}$$

Add on both sides $\sum_{D_i \in \mathcal{D}^w}(.) + \sum_{D_i \in \mathcal{D}^-}(.)$ to get

$$2\sum_{D_i \in \mathcal{D}^w} \ln\left(\frac{1}{\beta_i}\right) + \sum_{D_i \in \mathcal{D}^-} \ln\left(\frac{1}{\beta_i}\right) \;\geq\; \sum_{i=1}^{L} \ln\left(\frac{1}{\beta_i}\right) \tag{15}$$

Then add $\sum_{D_i \in \mathcal{D}^-}(.)$ on the left side of the inequality. For the inequality to hold, the added quantity should be positive. To guarantee this, we require that all the terms in the summation are nonnegative, i.e., $\ln\left(\frac{1}{\beta_i}\right) \geq 0$, which is equivalent to $\beta_i \leq 1$.

Then the lefthand side of (15) is twice the sum of all weights for the wrong classes, i.e.,

$$2\sum_{i=1}^{L} l_j^i \ln\left(\frac{1}{\beta_i}\right) \;\geq\; \sum_{i=1}^{L} \ln\left(\frac{1}{\beta_i}\right), \tag{16}$$

$$\sum_{i=1}^{L} \ln\left(\beta_i\right)^{-l_j^i} \quad \geq \quad \sum_{i=1}^{L} \ln\left(\beta_i\right)^{-\frac{1}{2}} \tag{17}$$

$$\prod_{i=1}^{L} \beta_i^{(1-l_j^i)} \quad \geq \quad \prod_{i=1}^{L} \beta_i^{\frac{1}{2}}. \tag{18}$$

Taking (13), (18) and (12) together,

$$1 \geq \sum_{\mathbf{z}_j \in \mathbf{Z}^{(-)}} w_j^1 \prod_{i=1}^{L} \frac{\beta_i^{(1-l_j^i)}}{C_i} \tag{19}$$

$$\geq \left(\sum_{\mathbf{z}_j \in \mathbf{Z}^{(-)}} w_j^1 \right) \prod_{i=1}^{L} \frac{\beta_i^{\frac{1}{2}}}{C_i} \quad = \quad \epsilon \cdot \prod_{i=1}^{L} \frac{\beta_i^{\frac{1}{2}}}{C_i}. \tag{20}$$

Solving for ϵ,

$$\epsilon \quad \leq \quad \prod_{i=1}^{L} \frac{C_i}{\beta_i^{\frac{1}{2}}}. \tag{21}$$

From the Lemma,

$$C_i = \sum_{k=1}^{N} w_k^i \beta_i^{(1-l_k^i)} \leq \sum_{k=1}^{N} w_k^i \left(1 - (1-\beta_i)(1-l_k^i)\right) \tag{22}$$

$$= \sum_{k=1}^{N} w_k^i \left(\beta_i + l_k^i - \beta_i l_k^i)\right) \tag{23}$$

$$= \beta_i \sum_{k=1}^{N} w_k^i + \sum_{k=1}^{N} w_k^i l_k^i - \beta_i \sum_{k=1}^{N} w_k^i l_k^i \tag{24}$$

$$= \beta_i + \epsilon_i - \beta_i \epsilon_i = 1 - (1-\beta_i)(1-\epsilon_i). \tag{25}$$

Combining (21) and (25)

$$\boxed{\epsilon \quad \leq \quad \prod_{i=1}^{L} \frac{1-(1-\beta_i)(1-\epsilon_i)}{\sqrt{\beta_i}}. \qquad (26)}$$

The next step is to find β_i's that minimize the bound of ϵ in (26). Setting the first derivative to zero and solving for β_i, we obtain

$$\boxed{\beta_i = \frac{\epsilon_i}{1-\epsilon_i}. \qquad (27)}$$

The second derivative of the righthand side at $\beta_i = \frac{\epsilon_i}{1-\epsilon_i}$ is positive, therefore the solution for β_i is a minimum of the bound. Substituting (27) into (26) leads to the thesis of the theorem

$$\boxed{\epsilon \;<\; 2^L \prod_{i=1}^{L} \sqrt{\epsilon_i(1-\epsilon_i)}. \qquad (28)}$$

$\blacksquare$

To illustrate the upper bound we generated a random sequence of individual errors $\epsilon_k \in (0, 0.5)$. for $L = 50$ classifiers. Plotted in Figure 2 is the average from 1000 such runs with one standard deviation on each side.

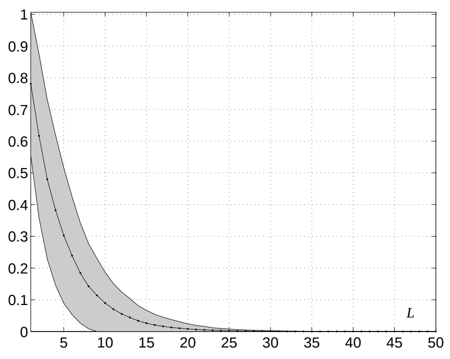

Fig. 2. A simulated upper bound of the training error of AdaBoost as a function of the number of classifier L and random individual errors in $(0, 0.5)$. The average of 1000 simulation runs is plotted with one standard deviation on each side.

Note that the ensemble error is practically 0 at $L = 30$. Hoping that the generalization error will follow a corresponding pattern, in many experimental studies AdaBoost is run up to $L = 50$ classifiers.

To gurarantee $\beta_i < 1$, AdaBoost re-initializes the weights to $\frac{1}{N}$ if $\epsilon_i \geq 0.5$. Freund and Schapire argue that having an error greater than half is too strict a demand for a multiple-class *weak learner* D_i. Even though the concern about the restriction being too severe is intuitive, we have to stress that ϵ_k is not the conventional error of classifier D_k. It is its *weighted* error. This means that if we applied D_k on a data set drawn from the problem in question, its (conventional) error could be quite different from ϵ_k, both ways: larger or smaller.

Theorem 2. (Aggressive) $\boxed{\xi(l_k^j) = 1 - 2l_k^j}$ *Let ϵ be the ensemble training error and let ϵ_i, $i = 1, \ldots, L$ be the weighted training errors of the classifiers in $\mathcal{D}$ as in (1). Then*

$$\epsilon \;\leq\; \prod_{i=1}^{L} \; 1 - (1-\beta_i)(1-2\epsilon_i). \qquad (29)$$

Proof. The proof matches that of Theorem 1 up to inequality (16). The only difference is that $\beta_i^{(1-l_j^i)}$ is replaced by $\beta_i^{(1-2l_j^i)}$. Adding $\sum_i \ln(\beta_i)$ on both sides of (16), and taking the exponent, we arrive at

$$\prod_{i=1}^{L} \beta_i^{(1-2l_j^i)} \quad \geq \quad 1. \tag{30}$$

From (20),

$$1 \geq \sum_{\mathbf{z}_j \in \mathbf{Z}^{(-)}} w_j^1 \prod_{i=1}^{L} \frac{\beta_i^{(1-2l_j^i)}}{C_i} \geq \left(\sum_{\mathbf{z}_j \in \mathbf{Z}^{(-)}} w_j^1 \right) \prod_{i=1}^{L} \frac{1}{C_i} = \epsilon \cdot \prod_{i=1}^{L} \frac{1}{C_i}. \tag{31}$$

Solving for ϵ,

$$\epsilon \quad \leq \quad \prod_{i=1}^{L} C_i. \tag{32}$$

Using the Lemma

$$\boxed{\epsilon \quad \leq \quad \prod_{i=1}^{L} 1 - (1-\beta_i)(1-2\epsilon_i). \qquad (33)}$$

■

The curious finding here is that the bound is linear on β_i. The first derivative is positive if we assume $\epsilon < 0.5$, therefore the smaller the β_i, the better the bound. We can solve

$$1 - (1-\beta_i)(1-2\epsilon_i) \quad \leq \quad 2\sqrt{\epsilon_i(1-\epsilon_i)} \tag{34}$$

for β_i to find out for which values the Aggressive bound is better than the Conservative.1 bound. If we restrict ϵ_i within $(0, 0.2)$ and use

$$\beta_i = \frac{\sqrt{\epsilon_i(1-\epsilon_i)} - 2\epsilon_i}{1-2\epsilon_i}, \qquad \text{(The restrcition guarantees } \beta > 0\text{)} \tag{35}$$

then we reduce the error bound of Conservative.1 by a factor of 2^L, i.e.,

$$\epsilon \quad < \quad \prod_{i=1}^{L} \sqrt{\epsilon_i(1-\epsilon_i)}. \tag{36}$$

Theorem 3. (Conservative.2) $\boxed{\xi(l_k^j) = -l_k^j}$ *Let ϵ be the ensemble training error and let ϵ_i, $i = 1, \ldots, L$ be the weighted training errors of the classifiers in $\mathcal{D}$ as in (1). Then for*

$$\beta_i = \frac{\epsilon_i(1-\epsilon_i)}{1+\epsilon_i}^2, \tag{37}$$

$$\epsilon \quad < \quad \prod_{i=1}^{L} \sqrt{\epsilon_i(1-\epsilon_i)}. \tag{38}$$

Proof. The proof follows these of Theorems 1 and 2 with $\beta_i^{(-l_j^i)}$ instead of $\beta_i^{(1-l_j^i)}$.

From (20) and (17),

$$1 \geq \sum_{\mathbf{z}_j \in \mathbf{Z}^{(-)}} w_j^1 \prod_{i=1}^{L} \frac{\beta_i^{(-l_j^i)}}{C_i} \geq \left(\sum_{\mathbf{z}_j \in \mathbf{Z}^{(-)}} w_j^1 \right) \prod_{i=1}^{L} \frac{\beta_i^{-\frac{1}{2}}}{C_i} = \epsilon \cdot \prod_{i=1}^{L} \frac{1}{C_i \sqrt{\beta_i}}. \qquad (39)$$

Solving for ϵ,

$$\epsilon \quad \leq \quad \prod_{i=1}^{L} C_i \sqrt{\beta_i}. \qquad (40)$$

Using the Lemma

$$\boxed{\epsilon \quad \leq \quad \prod_{i=1}^{L} \; (1 + \epsilon_i - \beta_i \epsilon_i) \sqrt{\beta_i} \qquad (41)}$$

The bound has a maximum for $\beta_i = \frac{1+\epsilon_i}{3\epsilon_i}$ but it is outside the range of interest (the interval $[0, 1]$) for β_i. If we pick β_i as in (37) and substitute into (41), the upper bound (38) is derived. As with the aggressive AdaBoost, this bound is better by a factor of 2^L and can be made arbitrarily better. ■

4 Comparison

Figure 3 illustrates the bounds for the three AdaBoost versions as functions of $\beta \in [0, 1]$. One term of the product is considered for each plot, so $\beta_i = \beta$, $\epsilon_i = \epsilon$, and U is the contribution of that term to the total error bound. ϵ was varied from 0 to 0.5. A gray line was plotted for each value of ϵ as a function $U(\beta)$. Do not be mislead by the scale: the most popular version (Conservative.1) has the worst bound. For reference we also plotted solid black lines for $\epsilon = \{0.05, 0.25, 0.45\}$. For the Conservative.1 version, the minima of U found by $\beta = \frac{\epsilon}{1-\epsilon}$ for the three values of ϵ are plotted as dots and encircled.

Figure 4 shows $U(\beta)$ for three values of ϵ. The bounds of the three AdaBoost versions are given on the same plot. The plots show that the conventional Conservative.1 version has the largest bound of the three AdaBoost variants.

5 Conclusions

Three AdaBoost variants are detailed and the error bounds on the training errors are derived (following the proof by Freund and Schapire [3] of the version called here Conservative.1). We found that the Aggressive and Conservative.2 versions have smaller error bounds. More importantly, whereas the bound $U(\beta)$ for Conservative.1 version has a unique minimum for $\beta = \frac{\epsilon}{1-\epsilon}$, the Aggressive and Conservative.2 AdaBoost are monotone for $\beta \in [0, 1]$. This gives infinitely

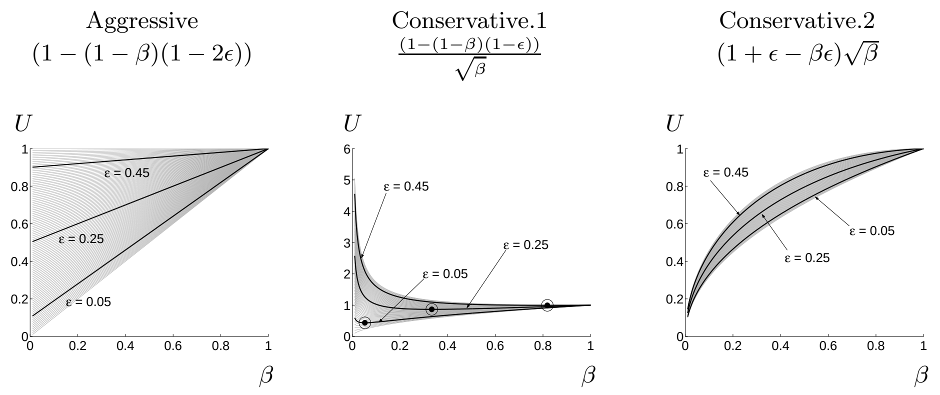

Fig. 3. Upper bounds on the training error for AdaBoost (one term in the product) for the three variants.

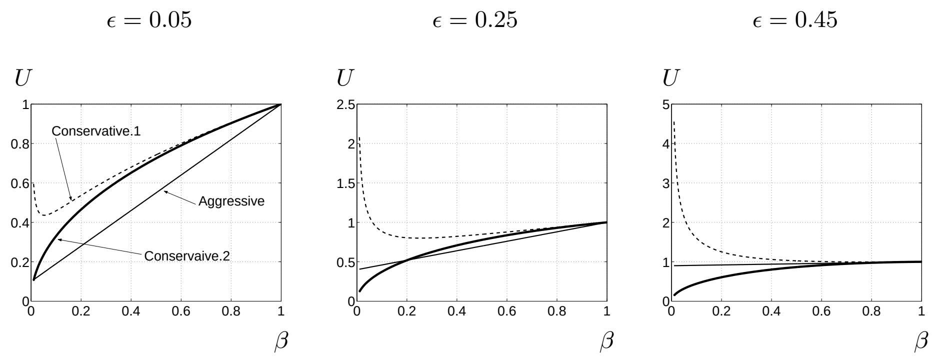

Fig. 4. Upper bounds on the training error for AdaBoost (one term in the product) for the three variants.

many choices for better error bounds than Conservative.1. Theorems 2 and 3 prove the bounds and give suggestions for β. An important point to be emphasized here is that the *training* error bounds do not guarantee anything about generalization. The results from this study that can be useful in real experiments are the suggestion for β (equations (35) and (37)) which guarantee training error bounds lower by a factor of 2^L than the training error bound of the conventional AdaBoost version (Conservative.1).

References

1. E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–142, 1999.
2. R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification.* John Wiley & Sons, NY, second edition, 2001.
3. Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
4. R.E. Schapire. Theoretical views of boosting. In *Proc. 4th European Conference on Computational Learning Theory*, pages 1–10, 1999.
5. R.E. Schapire. The boosting approach to machine learning. An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.