PREDICTION GAMES AND ARCING ALGORITHMS

Leo Breiman
leo@stat.berkeley.edu

Technical Report 504
December 19, 1997,
revised December 14,1998
Statistics Department
University of California
Berkeley, CA. (4720

Abstract

The theory behind the success of adaptive reweighting and combining algorithms (arcing) such as Adaboost (Freund and Schapire [1995, 1996a]) and others in reducing generalization error has not been well understood. By formulating prediction as a game where one player makes a selection from instances in the training set and the other a convex linear combination of predictors from a finite set, existing arcing algorithms are shown to be algorithms for finding good game strategies. The minimax theorem is an essential ingredient of the convergence proofs. An arcing algorithm is described that converges to the optimal strategy. A bound on the generalization error for the combined predictors in terms of their maximum error is proven that is sharper than bounds to date. Schapire et al. [1997] offered an explanation of why Adaboost works in terms of its ability to produce generally high margins. The empirical comparison of Adaboost to the optimal arcing algorithm shows that their explanation is not complete.

1. **Introduction**

Recent empirical work has shown that combining predictors can lead to significant reductions in generalization error. Interestingly, the individual predictors can be very simple, i.e. single hyperplanes in two class classification (Ji and Ma[1997]) or two terminal-node trees (stumps) Schapire et.al[1997]. While the empirical work has given exciting results, our full understanding of why it works is only partially filled in.

Let $\{h_m(\mathbf{x})\}$ be a set of M classifiers defined on input vectors $\mathbf{x}$ where $h_m(\mathbf{x})$ takes values in one of J class labels. Denote by $\{c_m\}$ an M-vector of constants such that $c_m \geq 0, \sum c_m = 1$. The combined classifiers predict that class having the plurality of the weighted votes. That is, the predicted class is

$$\arg\max_y \sum c_m I(h_m(\mathbf{x})=y)$$

where I(true)=1, I(false)=0, and y ranges over the set of class labels.

The problem is this: *given an N-instance training set* $T=\{(y_n,\mathbf{x}_n), n=1,...,N\}$ *and a set of M predictors* $\{h_m(\mathbf{x})\}$ *find* $\{c_m\}$ *such that the combined predictor has low generalization error.*

The approachs that have been very successful to date construct a sequence of altered training sets, find the predictor in the class that minimizes the training set error on the current altered training set, and use this information together with past information to construct the next altered data set. The weights $\{c_m\}$ are also determined in this sequential process.

1.1 Background

The first well known combination algorithm was bagging (Breiman[1996b]). The altered training sets were taken to be bootstrap samples from the original training set, and each predictor grown had equal weighting. It proved quite effective in reducing generalization error. The explanation given for its success was in terms of the bias-variance components of the generalization error. The variance is the scatter in the predictions gotten from using different training sets, each one drawn from the same distribution. Average all of these predictions (or take their most probable value in classification) and compute how much this average differs from the target function. The result is bias. Breiman[1996] shows that tree algorithms have small bias and the effect of combination is to reduce the variance.

Freund and Schapire[1996] introduced an combination algorithm called Adaboost which was designed to drive the training error rapidly to zero. But experiments showed that Adaboost kept lowering the generalization error long after the training set error was zero. The resulting generalization errors were significantly lower than those produced by bagging (Breiman[1996a]. Drucker and Cortes[1995], Quinlan [1996], Freund and Schapire[1996a], Kong and Dietterich[1996], Bauer and Kohavi[1998]).

The Adaboost algorithm differed significantly from bagging, and begged the question of why it worked as well as it did. Breiman[1996b] showed that for trees, Adaboost reduced variance more than bagging while keeping bias low. leading to the possible conclusion that it was a more effective variance reduction algorithm. But Schapire et al.[1997] gave examples of data where two-node trees (stumps) had high bias and the main effect of Adaboost was to reduce the bias.

Another explanation for Adaboost's success was offered in the Schapire et al.[1997] paper. For any combination of classifiers with non-negative weights $\mathbf{c}=\{c_m\}$ summing to one, define the margin $mg(\mathbf{z},\mathbf{c})$ at input $\mathbf{z}=(y,\mathbf{x})$ as

$$mg(\mathbf{z},\mathbf{c})=\sum c_m I(h_m(\mathbf{x})=y)-\max_{y'\neq y}\sum c_m I(h_m(\mathbf{x})=y'). \qquad (1.1)$$

Thus, the margin is the total vote for the correct class minus the total vote for the next highest class. Intuitively, if the margins over a training set are generally high, then the misclassifications, corresponding to all test set inputs such that $mg(\mathbf{z},\mathbf{c})<0$, will be low. In general, if $\mathbf{Z}=(Y,\mathbf{X})$ is a random vector selected from the same distribution as the instances in T, but independent of them, the generalization error is $P(mg(\mathbf{Z},\mathbf{c})<0)$

Schapire et al. [1997] derived a bound on the generalization error of a combination of classifiers that did not depend on how many classifiers were combined, but only on the training set margin distribution, the sample size and VC-dimension of the set of classifiers. Then they showed, experimentally, that Adaboost produced generally higher margins than bagging. They draw the conclusion from this that the higher the margins (all else being equal) the lower the generalization error, and implied that the key to the success of Adaboost was its ability to produce large margins.

Meanwhile other combination algorithms, differing from Adaboost, have been explored. One was arc-x4, which Breiman[1996b] showed had error performance comparable to Adaboost. Another was an algorithm that used hyperplanes as the class of predictors and produced low error on some hard problems (Ji and Ma[1997]). All three algorithms had the common property that the current altered training set weighted more heavily examples that had been frequently misclassified in the past. But any more precise statement of what they have in common has been lacking.

1.2 Outline of Results

Replacing the maximum in (1.1) by a sum gives

$$mg(\mathbf{z}_n,\mathbf{c})\geq 1-2\sum_n c_m I(y_n \neq h_m(\mathbf{x}_n)) \tag{1.2}$$

with equality in the two class situation. Denote

$$er(\mathbf{z},\mathbf{c})=\sum c_m I(y\neq h_m(\mathbf{x})) \tag{1.3}$$

so that $mg(\mathbf{z}_n,\mathbf{c})\geq 1-2er(\mathbf{z}_n,\mathbf{c})$. Now $er(\mathbf{z},\mathbf{c})$ is the $\{c_m\}$ weighted frequency of misclassifications over the set of predictors $\{h_m(\mathbf{x})\}$. The smaller we can make $er(\mathbf{z},\mathbf{c})$, the larger the margins. In particular, define:

$$top(\mathbf{c})=\max_{\mathbf{z}\varepsilon T} er(\mathbf{z},\mathbf{c}) \tag{1.4}$$

Then,

$$\min_{\mathbf{z}\varepsilon T} mg(\mathbf{z},\mathbf{c})\geq -2top(\mathbf{c})+1 \tag{1.5}$$

The smaller $top(\mathbf{c})$, the larger the minimum value of the margin.

In Section 2 a game theoretic context is introduced and the minimax theorem gives a fundamental relation between the maximum value of $top(\mathbf{c})$ over all values of $\mathbf{c}$ and other parameters of the problems. This relation will be critical in our convergence proofs for arcing algorithms.

In Section 3 we define a computationally feasible class of algorithms for producing generally low values of $er(\mathbf{z},\mathbf{c})$. These are called arcing algorithms--an acronym for **A**daptive **R**eweighting and **C**ombin**ing.** Adaboost, arc-x4 and random hyperplanes are defined as examples of arcing algorithms.

Section 4 discusses the convergence of arcing algorithms. Two types of arcing algorithms are defined. We prove that the iterations in both types converge to low values of $top(\mathbf{c})$ or to low average values of a specified function of $er(\mathbf{z},\mathbf{c})$. A critical element in these proofs is the min-max relation. It's shown that Adaboost belongs to one type--arc-x4 and random hyperplanes to another. These results give the unifying thread between the various successful combination algorithms--they are all arcing methods for producing low values of $top(\mathbf{c})$ or some functional average of $er(\mathbf{z},\mathbf{c})$.

Section 5 defines an arcing algorithm called arc-gv and proves that under arc-gv the values of $top(\mathbf{c}^{(k)})$ converge to the lowest possible value of $top(\mathbf{c})$. Section 6 gives an upper bound for the generalization error of a combination of classifiers in terms of $top(\mathbf{c})$, the sample size of the training set, and (essentially) the VC-dimension of the predictors in the class $\{h_m\}$.

The bound is sharper than the bound in Schapire et.al[1997] based on the margin distribution but uses the same elegant device in its derivation. If the Schapire et al. bound implies that the margin distribution is the key to the generalization error, the bound in terms of $top(\mathbf{c})$ implies even more strongly that $top(\mathbf{c})$ is the key to the generalization error.

This is followed in Section 7 by experimental results applying arc-gv and Adaboost to various data sets using tree classifiers confined to a specified number of terminal nodes in order to fix their VC-dimension. The surprise is that even though arc-gv produces lower values of $top(\mathbf{c})$ than Adaboost, its test set error is higher. We also show that the margin distributions using arc-gv dominate those gotten by using Adaboost--i.e. arc-gv produces generally higher margins.

Section 8 gives surmises and hopes. It seems that simply producing larger margins or lower tops while keeping the VC-dimension fixed does not imply lower generalization error.

Lengthy or difficult proofs are banished to the Appendices.

2. The Prediction Game

One way to formulate the idea that $er(\mathbf{z},\mathbf{c})$ is generally small for $\mathbf{z}\varepsilon T$ is by requiring uniform smallness.

Definition 2.1 *Define the function top(c) on M-vectors $\{c_m\}$ as*

$$top(\mathbf{c})=\max_n er(\mathbf{z}_n,\mathbf{c})$$

Two questions are involved in making $top(\mathbf{c})$ small:

i) What is the value of $\inf_{\mathbf{c}} top(\mathbf{c})$?

ii) What are effective algorithms for producing small values of $top(\mathbf{c})$?

By formulating combinations in terms of a prediction game, we will see that these two questions are linked.

<u>Definition 2.2</u> *The prediction game is a two player zero-sum matrix game. Player I chooses* $\mathbf{z}_n \varepsilon T$. *Player II chooses* $\{c_m\}$. *Player I wins the amount* $er(\mathbf{z}_n,\mathbf{c})$

Now $er(\mathbf{z}_n,\mathbf{c})$ is continuous and linear in $\mathbf{c}$ for each $\mathbf{z}_n \varepsilon T$ fixed. By standard game theory results (Blackwell and Girshick 1954]), Player II has a good pure strategy, Player I has a good mixed strategy (a probability measure on the instances in T) and the value of the game ϕ^* is given by the minimax theorem:

$$\phi^* = \inf_{\mathbf{c}} \sup_Q E_Q er(\mathbf{z},\mathbf{c}) = \sup_Q \inf_{\mathbf{c}} E_Q er(\mathbf{z},\mathbf{c}) \qquad (2.1)$$

where the Q are probability measures on the instances in T. Note that

$$top(\mathbf{c}) = \sup_Q E_Q er(\mathbf{z},\mathbf{c}).$$

Then defining $e_m = \{\mathbf{z}_n : y_n \neq h_m(\mathbf{x}_n)\}$ as the error set of the mth predictor, rewrite (2.1) as

$$\phi^* = \inf_{\mathbf{c}} top(\mathbf{c}) = \sup_Q \min_m Q(e_m) \qquad (2.2)$$

The equation (2.2) is the key to our analysis of arcing algorithms. Relation (2.2) also follows from the duality theorem of linear programming (Breiman[1997]). The classification game was introduced in Freund and Schapire[1996b].

3. <u>Arcing Algorithms</u>

The algorithmic problem is how to determine $\mathbf{c}$ so that $er(\mathbf{z},\mathbf{c})$ is generally small for $\mathbf{z}$ in T. This can be formulated in different ways as a minimization problem to which standard optimization methods can be applied. For instance, linear programming methods can be used to find a $\mathbf{c}$ such that $\phi^* = top(\mathbf{c})$. But such methods are not feasible in practice. Typically, the set of classifiers is large and complex. It would be extremely difficult to work with many at a time as required by standard optimization methods.

3.1 Definition of Arcing Algorithms

The essence of feasible algorithms is that it is possible to solve, in practice, problems of this following type:

Weighted Minimization. *Given any probability weighting* $Q(\mathbf{z}_n)$ *on the instances in T, find the predictor in the set* $\{h_m\}$ *minimizing* $Q(e_m)$.

This means, minimize the Q-weighted misclassification rate. This is not always exactly possible. For example, the CART algorithm does not find that J-terminal node tree having minimum Q-weighted error. Instead, it uses a greedy algorithm to approximate the minimizing tree. In the theory below, we will assume that it is possible to find the minimizing h_m, keeping in mind that this may be only approximately true in practice.

Definition 3.1 *An arcing algorithm works on a vector* $\mathbf{b}$ *of non-negative weights such that* b_m *is the weight assigned to predictor* h_m *and the* $\mathbf{c}$ *vector is given by* $\mathbf{b}/|\mathbf{b}|$, *where* $|\mathbf{b}|=\sum b_m$. *The algorithm updates* $\mathbf{b}$ *in these steps:*

i) Depending on the outcomes of the first k steps, a probability weight Q is constructed on T.
ii) The (k+1)st predictor selected is that h_m *minimizing* $Q(e_m)$.
iii) Increase b_m *for the minimizing value of m. The amount of the increase depends on the first k+1 predictors selected.*
iv) Repeat until satisfactory convergence.

Arcing is an acronym for **A**daptive **R**eweighting and **C**ombin**ing** (Breiman[1997]). Each step in an arcing algorithm consists of a weighted minimization followed by a recomputation of $\mathbf{c}$ and Q. The usual initial values are $\mathbf{b}=0$ and Q uniform over T.

In the following sections we will give examples of arcing algorithms together with general descriptions and convergence properties.

3.2 **Examples of Arcing Algorithms,**

There are a number of successful arcing algorithms appearing in recent literature. We give three examples that have given excellent empirical performance in terms of generalization error.

Example 1. Adaboost (Freund and Schapire ([1995], [1996a]).

If h_m is selected at the kth step, compute $\varepsilon_k = Q_k(e_m)$. Let $\beta_k = (1-\varepsilon_k)/\varepsilon_k$, denote $l_m(\mathbf{z}_n) = I(y_n \neq h_m(\mathbf{x}_n))$ and update by:

$$Q_{k+1}(\mathbf{z}_n) = Q_k(\mathbf{z}_n)\beta_k^{\,l_m(\mathbf{z}_n)}/S$$

where the /S indicates normalization to sum one. Put b_m equal to $\log(\beta_k)$.

<u>Example 2.</u> Arc-x4 (Breiman [1997])

Define $ms_{(k)}(\mathbf{z}_n)$ as the number of misclassifications of $\mathbf{x}_n$ by the first k classifiers selected. Let

$$Q_{k+1}(\mathbf{z}_n) = (1 + (ms_{(k)}(\mathbf{z}_n))^4)/S$$

If h_m is selected at the kth step, put b_m equal to one.

<u>Example 3.</u> Random Hyperplanes (Ji and Ma[1997])

This method applies only to two-class problems. The set of classifiers H is defined this way: Each vector in input space and point $\mathbf{x}_n$ in the training set defines two classifiers. Form the hyperplane passing through $\mathbf{x}_n$ perpendicular to the given vector. The first classifier classifies all the points on one side as class 1 and on the other as class 2. The second classifier switches the class assignment.

Set two parameters $\alpha > 0$, $\eta > 0$ such that $.5 - \eta < \alpha < .5$. After the kth classifier is selected, set its b weight to equal one. Let

$$Q_{k+1}(\mathbf{z}_n) = I(ms_{(k)}(\mathbf{z}_n) > \alpha k)/S$$

where $I(\cdot)$ is the indicator function. Select a hyperplane direction and training set instance at random. Compute the classification error for each of the two associated classifiers using the probabilities $Q_{k+1}(\mathbf{z}_n)$. If the smaller of the two errors is less than $.5 - \eta$ then keep the corresponding classifier. Otherwise, reject both and select another random hyperplane.

4 Convergence of Arcing Algorithms

The interesting question is: do arcing algorithms converge, and if so, what do they converge to? More explicitly, arcing algorithms generate sequences $\{\mathbf{c}^{(k)}\}$ of normalized weight vectors. What can be said about the convergence of the values of $er(\mathbf{z}_n,\mathbf{c}^{(k)})$ or of $top(\mathbf{c}^{(k)})$?

The results below place arcing algorithms into a unifying context of numerical analysis concerned with the convergence of optimization algorithms. Arcing algorithms become simply iterative methods for optimizing some criteria involving the values of $er(\mathbf{z},\mathbf{c})$. But it will be interesting to find out what the algorithms are optimizing--for instance, what is arc-x4 optimizing? Also important is the fact that they converge to the optimal value.

Inspection of Adaboost and of arc-x4 and random hyperplanes shows that the algorithms involved are of two intrinsically different types. Adaboost is in one type and arc-x4 and random hyperplanes are in the second type. The first type define a function $g(\mathbf{b})$ of the unnormalized weights **b,** iteratively minimizes $g(\mathbf{b})$ and in the process reduces the values of $er(\mathbf{z}_n,\mathbf{c})$. The second type work directly on minimizing a function $g(\mathbf{c})$ of the normalized weights. All existing arcing algorithms (see also Leisch and Hornik[1997]) fall into one of these two types.

4.1 Type I Arcing Algorithms.

Let $f(x)$ be any function of a single real variable defined on the whole line such that $f(x)\to\infty$ as $x\to\infty$, to 0 as $x\to-\infty$ with everywhere positive first and second derivatives. For weights $\{b_m\}$ we slightly abuse notation and set $er(\mathbf{z}_n,\mathbf{b})=\sum_m b_m l_m(\mathbf{z}_n)$ where $l_m(\mathbf{z}_n)=I(y_n\neq h_m(\mathbf{x}_n))$. Assuming that $\phi>\phi^*$, consider minimizing $g(\mathbf{b})=\sum_n f(er(\mathbf{z}_n,\mathbf{b})-\phi|\mathbf{b}|)$ starting from $\mathbf{b}=0$.

Definition 4.1 *A Type I arcing algorithm updates* ***b*** *as follows: At the current value of* ***b*** *let*

$$Q(\mathbf{z}_n)=f'(er(\mathbf{z}_n,\mathbf{b}))-\phi|\mathbf{b}|)/S$$

and $m^*=\mathrm{argmin}_m Q(e_m)$. *Add* $\Delta>0$ *to* b_{m^*} *and do a line search to minimize* $g(\mathbf{b}+\Delta\mathbf{u}_{m^*})$ *over* $\Delta>0$ *where* $\mathbf{u}_{m^*}$ *is a unit vector in the* *direction of* b_{m*}. *If the minimizing value of* Δ *is* Δ^* *then update* $\mathbf{b}\to\mathbf{b}+\Delta^*\mathbf{u}_{m*}$. *Repeat until convergence.*

Comments. Note that

$$\partial g(\mathbf{b})/\partial b_m=(E_Q(e_m)-\phi)\sum_n f'(er(\mathbf{z}_n,\mathbf{c})-\phi|\mathbf{b}|)$$

so the minimum value of the first partial of the target function g(**b**) is in the direction of b_{m*}. This value is negative, because by the minimax theorem $\min_m E_Q(e_m)\le\phi^*$. Furthermore, the 2nd derivative of $g(\mathbf{b}+\Delta\mathbf{u}_{m*})$ with respect to Δ is positive, insuring a unique minimum in the line search over $\Delta>0$.

Theorem 4.2. *Let* $\mathbf{b}^{(k)}$ *be the successive values generated by a Type I arcing algorithm, and set* $\mathbf{c}^{(k)}=\mathbf{b}^{(k)}/|\mathbf{b}|$ *Then* $\limsup_k top(\mathbf{c}^{(k)})\le\phi$

proof. See Appendix A

Proposition 4.3. *Adaboost is a Type I arcing algorithm using* $f(x)=e^x$ *and* $\phi=1/2$.

Proof: For $f(x)=e^x$

$$f(er(\mathbf{z}_n,\mathbf{b})-\phi|\mathbf{b}|)=e^{-\phi|\mathbf{b}|}\prod_m e^{b_m l_m(\mathbf{z}_n)}$$

Denote $\pi(\mathbf{z}_n)=\prod_m \exp(b_m l_m(\mathbf{z}_n))$. Set $Q(\mathbf{z}_n)=\pi(\mathbf{z}_n)/\sum_h \pi(\mathbf{z}_h)$, m*= $\text{argmin}_m\, Q(e_m)$. Set $\varepsilon_m=Q(e_{m*})$. We do the line search step by solving

$$\sum_n (l_m(\mathbf{z}_n)-\phi)\, f'(er(\mathbf{z}_n,\mathbf{b}+\Delta\mathbf{u}_{m*})-\phi|\mathbf{b}|-\phi\Delta)=0$$

which gives $\Delta^*=\log(\phi/(1-\phi))+\log((1-\varepsilon_m)/\varepsilon_m)$. The update for Q is given in terms of

$$\pi(\mathbf{z}_n)\to\pi(\mathbf{z}_n)e^{\Delta^* l_m(\mathbf{z}_n)}/S.$$

For $\phi=1/2$ this is the Adaboost algorithm described in Section 1.

Schapire et al.[1997] note that the Adaboost algorithm produces a **c** sequence so that $\limsup_{\mathbf{c}} top(\mathbf{c}) = \phi_1$ where ϕ_1 is less than 1/ 2. In fact, we can show that

$$\limsup_{\mathbf{c}} top(\mathbf{c}) \le (\log 2 + \log(1-\phi^*))/(-\log(\phi^*) + \log(1-\phi^*)).$$

If, for instance, $\phi^*=.25$, this bound equals .37. Thus, even though Theorem 4.2 only guarantees an upper bound of .5, using the exponential form of f allows a sharper bound to be given.

4.2 Type II Arcing Algorithms

A Type II algorithm minimizes $g(\mathbf{c}) = \sum_n f(er(\mathbf{z}_n, \mathbf{c}))$ where f'(x) is non-negative and f''(x) is continuous and non-negative for all x in the interval [0,1]. Unlike the Type I algorithms which aim directly at producing low values of $top(\mathbf{c})$, the Type II algorithms produce $\inf_{\mathbf{c}} Ef(\mathbf{z},\mathbf{c})$ where the expectation E is with respect to the uniform distribution on T. Thus, it tries to get generally, but not uniformly, small values of $er(\mathbf{z}_n, \mathbf{c})$

Definition 4.4 *Let* $\mathbf{c} = \mathbf{b}/|\mathbf{b}|$. *A Type II arcing algorithm updates* ***b*** *as follows: At the current value of* ***b*** *, if* $\sum_n f'(er(\mathbf{z}_n, \mathbf{c})) = 0$ *then stop. Otherwise, let*

$$Q(\mathbf{z}_n) = f'(er(\mathbf{z}_n, \mathbf{c}))/S$$

and $m^* = \arg\min_m Q(e_m)\cdot$ *If* $Q(e_{m^*}) \ge E_Q(er(\mathbf{z},\mathbf{c}))$ *then stop. Otherwise let* $b_{m^*} = b_{m^*} + 1$ *and repeat.*

Comment: Since

$$\partial g(\mathbf{c})/\partial b_m = \frac{1}{|\mathbf{b}|} \sum_n (l_m(\mathbf{z}_n) - er(\mathbf{z}_n, \mathbf{c})) f'(er(\mathbf{z}_n, \mathbf{c})),$$

the smallest partial derivative is at m=m*.

Theorem 4.5 *Let* ***c*** *be any stopping or limit point of a Type II arcing algorithm. Then* ***c*** *is a global minimum of* $g(\mathbf{c})$*,*

Proof: see Appendix B.

Proposition 4.6 *Arc-x4 is a Type II arcing algorithm*

Proof. In Type II arcing, the b-weight of each minimizing classifier is one, or an integer greater than one if a classifier minimizes $Q(e_m)$ repeatedly. Hence, the proportion of misclassifications of $\mathbf{z}_n$ is $er(\mathbf{z}_n,\mathbf{c})$. At each stage in arc-x4, the current probability $Q(\mathbf{z}_n)$ is taken proportional to $er(\mathbf{z}_n,\mathbf{c})^4$. Hence, the arc-x4 algorithm is minimizing $\sum_n er(\mathbf{z}_n,\mathbf{c})^5$.

There is a modified version of the Type II algorithm that works on getting low values of $top(\mathbf{c})$. Starts with a function $q(x)$ defined on [-1,1] such that $q'(x)$ is zero for $x \le 0$, positive for $x>0$, and q'' continuous, bounded and non-negative. For $\phi>\phi^*$ define $f(x)=q(x-\phi)$. Applying Theorem 4.5 to this function gives the following result:

<u>Corollary 4.7</u> *For* $\mathbf{c}$ *any limit or stopping point of a Type II algorithm, using* $f(x)=q(x-\phi)$, $top(\mathbf{c})\le\phi$

<u>Proof:</u> $top(\mathbf{c})\le\phi$ is necessary and sufficient for a global minimum of g($\mathbf{c}$).

<u>Proposition 4.8</u> *Random Hyperplanes is (almost) a Type II arcing algorithm.*

<u>Proof.</u> Take $\phi>\phi^*$, $q(x)=x^+$, $f(x)=q(x-\phi)$ and consider trying to minimize $\sum_n f(er(\mathbf{z}_n,\mathbf{c}))$ using the Type II algorithm. At each stage, the current probability $Q(\mathbf{z}_n)$ is proportional to $I(er(\mathbf{z}_n,\mathbf{c})-\phi>0)$ where I is the indicator function, and this is the Ji-Ma reweighting. In the standard form of the Type II algorithm, e_{m^*} minimizes $Q(e_m)$ and the corresponding b value is increased by one. Because x^+ does not have a bounded 2nd derivative and because the Ji-Ma algorithm does only a restricted search for the minimizing e_m, the Type II arcing algorithm has to be modified a bit to make the convergence proof work.

Take $\varepsilon>0$ small, and $q(x)$ defined on [-1,1] such that $q'(x)=0$ for $x\le 0$, $q'(x)=1$ for $x>\varepsilon$, and $q'(x)$ in $[0,\varepsilon]$ rising smoothly from 0 to 1 so that $q''(x)$ is continuous, bounded and non-negative on [-1,1]. Now let $\phi>\phi^*$ and consider minimizing $\sum_n q(er(\mathbf{z}_n,\mathbf{c})-\phi)$. Take $\delta>0$ and at each stage, search randomly to find a classifier h_m such that $Q(e_m)\le\phi^*+\delta$. Then as long as $\phi^*+\delta-\phi<0$, the result of Corollary 4.7 holds.

The original Ji-Ma algorithm sets the values of two parameters $\alpha > 0, \eta > 0$. In our notation $\phi = \alpha, \phi^* + \delta = .5 - \eta$. Ji and Ma set the values of α, η by an experimental search. This is not surprising since the value of ϕ^* is unknown.

5 An Optimizing Arcing Algorithm

None of the arcing algorithms described above have the property that they drive $top(\mathbf{c})$ to its lowest possible value ϕ^*=the value of the game. This section describes an arcing algorithm we call arc-gv (gv=game value) and proves that $top(\mathbf{c}^{(k)}) \to \phi^*$. The algorithm generates a sequence of weight vectors $\mathbf{b}^{(k)}$ and normed weights $\mathbf{c}^{(k)} = \mathbf{b}^{(k)} / |\mathbf{b}^{(k)}|$ Denote $t_k = top(\mathbf{c}^{(k)})$. Initialize by taking $\mathbf{b}^{(1)} = 0$ and $Q_1(\mathbf{z}_n) = 1/N$, all n.

Definition 5.1 *Arc-gv updates* $\mathbf{b}^{(k)}$ *to* $\mathbf{b}^{(k+1)}$ *as follows:*

i) Let

$$Q_k(\mathbf{z}_n) = \exp(er(\mathbf{z}_n, \mathbf{b}^{(k)}) - t_k |\mathbf{b}^{(k)}|) / S$$

$$m_{k+1} = \arg\min_m Q_k(e_m)$$

ii) Let Δ_k *be the minimizer of*

$$E_{Q_k}(\exp(\Delta(l_{m_{k+1}}(\mathbf{z}) - t_k)))$$

in the range $[0,1]$.

iii) *If* $\Delta_k = 0$ *then stop. Otherwise increase the* m_{k+1} *st coordinate of* $\mathbf{b}^{(k)}$ *by the amount* Δ_k *to get* $\mathbf{b}^{(k+1)}$.

Theorem 5.2 *If arc-gv stops at the kth step, then* $top(\mathbf{c}^{(k)}) = \phi^*$. *If it does not stop at any finite step, then* $\lim_k top(\mathbf{c}^{(k)}) = \phi^*$.

Proof. see Appendix C which also shows that the minimizing Δ at the kth step is given by the simple expression:

$$\Delta=\log[\frac{t}{1-t}\frac{1-q}{q}]$$

where $q=Q_k(e_m)$ and t=$top(\mathbf{c}^{(k)})$.

There is an early sequential method for finding optimal strategies in matrix games known as the "method of fictitious play". Its convergence was proved by Robinson[1951]. A more accessible reference is Szep and Forgo [1985]. It is an arcing algorithm, but appears considerably less efficient than the arc-gv method.

6. A Bound on the Generalization Error

Schapire et.al[1997] derived a bound on the classification generalization error in terms of the distribution of $mg(\mathbf{z},\mathbf{c})$ on the instances of T. Using the same elegant device that they created, we derive a sharper bound using the value of $top(\mathbf{c})$ instead of the margin distribution.

Let **Z** = (Y,**X**) be a random vector having the same distribution that the instances in T were drawn from but independent of T and denote $er(\mathbf{Z},\mathbf{c})=\sum_m c_m l_m(\mathbf{Z})$. Define $\tilde{P}$ as the probability on the set of all N-instance training sets such that each one is drawn from the distribution P. Set $\delta>0$ Then:

Theorem 6.1 *For* $\Delta>0$ *, define*

$$R=\frac{8\log(2M)}{N\Delta^2}$$

Except for a set of training sets with $\tilde{P}$ *probability* $\leq\delta$*, for every* $\Delta\geq\sqrt{8/M}$ *and* **c**

$$P(er(\mathbf{Z},\mathbf{c})\geq\Delta+top(\mathbf{c}))\leq R(1+\log(1/R)+\log(2N))+(\log(M)/\delta)/N \qquad (6.1)$$

Proof is patterned after the Schapire et al.[1997]proof. See Appendix D.

Using the inequality $mg(\mathbf{Z},\mathbf{c})\geq 1-2er(\mathbf{Z},\mathbf{c})$ and setting $\Delta=1/2-top(\mathbf{c})$ gives:

$$P(mg(\mathbf{Z},\mathbf{c})\leq 0)\leq R(1+\log(1/R)+\log(2N))+(\log(M)/\delta)/N \qquad (6.2)$$

The bound in Schapire et al.[1997] depends on $P_T(mg(\mathbf{z},\mathbf{c})\leq\theta)$ where P_T is the uniform distribution over the training set and θ can be varied. If θ is taken to be the minimum value of the margin over the training set, then in the two-class case, their bound is about square root of the bound in (6.2). If the bound is non-trivial and <1, then (6.2) is less then the Schapire et al. bound.

The additional sharpness comes from using the uniform bound given by $top(\mathbf{c})$. We give this theorem and its proof mainly as a factor hopefully pointing in the right direction. Generalization to infinite sets of predictors can be given in terms of their VC-dimension (see Schapire et al.[1997]).

The motivation for proving this theorem is partly the following--Schapire et al. draw the conclusion from their bound that for a fixed set of predictors, the margin distribution governs the generalization error. One could just as well say that Theorem 6.1 and equation (6.2) shows that it is the value of $top(\mathbf{c})$ that governs the generalization error. But both bounds are greater than one in all practical cases, leaving ample room for other factors to influence the true generalization error.

7 Empirical results

Schapire et.al interpret their VC-type bound to mean that, all else being equal, higher margins result in lower generalization error. The bound in Section 6 could be similarly interpreted as, all else being equal, lower values of $top(\mathbf{c})$ result in lower generalization error.

To do an empirical check, we implemented an algorithm into CART which selects the minimum training set cost subtree having k terminal nodes, where k is user specified. More specifically, a tree is grown which splits down to one instance per terminal node, using the current weights on each instance to determine the splitting criterion. Then the algorithm determines which subtree having k terminal nodes has minimal weighted misclassification cost. Setting the trees selected to have k terminal nodes fixes the VC-dimension.

For fixed k, we compare Adaboost to arc-gv. The latter algorithm reduces top($\mathbf{c}$) to its minimum value, hence makes the margins generally large. Adaboost is not touted to do a maximal enlargement of the margins, hence should not, by theory to date, produce as low as generalization error as arc-gv. To check, we ran both algorithms on a variety of synthetic and real data sets varying the value of k. We restrict attention to two-class problems, where $mg(\mathbf{z}_n,\mathbf{c})=1-2er(\mathbf{z}_n,\mathbf{c})$.

In the three synthetic data sets used, training sets of size 300 and test sets of size 3000 were generated. After the algorithms were run for 100 iterations, the test sets were used to estimate the generalization error. With the real data sets, a random 10% of the instances were set aside and used as a test set. In both cases, the procedure was repeated 10 times and the test set results averaged. In each run, we kept track of top(**c**) and these values were also averaged over the 10 runs for each algorithm.

The synthetic data sets are called twonorm, threenorm, and ringnorm and are described in Breiman[1997]. The real data sets are all in the UCI repository. The real data sets have the following number of input variables and instances--breast cancer 9-699: ionosphere 34-351: sonar 60-208. Two values of k were used for each data set. One value was set low, and the other higher. Larger values of k were used for largest data set (breast cancer) so that tree sizes would be appropriate to the data set.

7.1 Test Set Error and top(**c**)

Table 1 Test Set Error(%) and Top(**c**) (x100)

	Test Set Error		Top(**c**)	
data set	arc-gv	Adaboost	arc-gv	Adaboost
twonorm				
k=8	5.3 .	4.9	21.5	23.5
k=16	6.0	4.9	10.7	13.8
threenorm				
k=8	18.6	17.9	32.5	33.5
k=16	18.5	17.8	21.7	24.7
ringnorm				
k=8	6.1	5.4	23.9	26.1
k=16	8.3	6.3	10.5	15.6
breast cancer				
k=16	3.3	2.9	20.7	22.2
k=32	3.4	2.7	11.8	13.6
ionosphere				
k=8	3.7	5.1	23.1	25.1
k=16	3.1	3.1	10.3	12.9
sonar				
k=8	11.9	8.1	11.4	12.4
k=16	16.7	14.3	8.0	12.7

Although the test set errors for arc-gv and Adaboost are generally close, the pattern is that Adaboost has a test set error less than that of arc-gv. On the other hand, top(**c**) is often significantly less for arc-gv than for Adaboost. But this does not translate into a lower test set error for arc-gv. Often, quite the contrary.

7.2 The Margin Distributions

A last question is whether lower values of top(**c**) translate into generally higher values of the margin. We looked at this by computing two cumulative distribution functions of $mg(\mathbf{z}_n,\mathbf{c})$ for each data set--one using the Adaboost values and the other, the arc-gv values. For the real data sets, the entire data set was used in the comparison. For the synthetic data, the first data set generated was used. In all cases, the larger number of terminal nodes (16 or 32) was used. The two distribution functions are compared in Figure 1 for the synthetic data sets and in Figure 2 for the real data sets. To compare the results with Table 1, recall that the minimum margin is one minus twice top(**c**).

In all cases the distribution of the margin under Adaboost is uniformly smaller that the distribution under arc-gv. The conclusion is that these different margin distributions, keeping the VC-dimension fixed, had little effect on the generalization error. In fact, that smaller margins were usually associated to smaller generalization error. These empirical results give a definitive negative vote as to whether the margin distribution or the value of top(**c**) determines the generalization error and casts doubt on the ability of the loose VC-type bounds to uncover the mechanism leading to low generalization error..

8 Remarks

The results above leave us in a quandary. The laboratory results for various arcing algorithms are excellent, but the theory is in disarray. The evidence is that if we try too hard to make the margins larger, then overfitting sets in. One possibility is that the VC-type bounds do not completely reflect the capacity of the set of classifiers. For interesting recent work in this direction see Golea et al.[1998] and Freund[1998]. My sense of it is that we just do not understand enough about what is going on.

9. Acknowledgments

This work has important seeds in the Schapire et al. 1997 paper and thought-provoking talks with Yoav Freund at the Newton Institute, Cambridge University, during the summer of 1997. A trio of hardworking and long suffering referees have my thanks for forcing me to produce a more readable paper.

References

Bauer, E. and Kohavi, R.[1998]An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting and Variants, Machine Learning 1-33.

Breiman, L. [1996a] Bias, Variance, and Arcing Classifiers, Technical Report 460, Statistics Department, University of California (available at www.stat.berkeley.edu)

Breiman, L. [1996b] Bagging predictors , Machine Learning 26, No. 2, pp. 123-140

Breiman, L. [1997] Arcing the Edge, Technical Report 486, Statistics Department, University of California (available at www.stat.berkeley.edu)

Drucker, H. and Cortes, C. [1995] Boosting decision trees, Advances in Neural Information Processing Systems Vol. 8, pp. 479-485.

Freund, Y.[1998] Self Bounding Learning Algorithms, (available at http://www.research.att.com/~yoav look under "publications".)

Freund, Y. and Schapire, R. [1995] A decision-theoretic generalization of on-line learning and an application to boosting. to appear , Journal of Computer and System Sciences.

Freund, Y. and Schapire, R. [1996a] Experiments with a new boosting algorithm, Machine Learning: Proceedings of the Thirteenth International Conference, pp. 148-156

Freund, Y. and Schapire, R. [1996b] Game Theory, On-line Prediction and Boosting, Proceedings of the 9th Annual Conference on Computational Learning Theory

Golea,M., Bartlett, P., Lee, W., and Mason, L.[1998] Generalization in Decision Trees and DNF: Does Size Matter? Advances in Neural Information Processing Systems, Vol. 10 pp. 259-265

Ji, C. and Ma, S. [1997] Combinations of weak classifiers, Special Issue of Neural Networks and Pattern Recognition, IEEE Trans. Neural Networks, Vol. 8, pp. 32-42

Kong, E. and Dietterich, T.,[1996] Error-correcting output coding corrects bias and variance, Proceedings of the Twelfth International Conference on Machine Learning, 313-321

Leisch, F and Hornik, K.[1997] ARC-LH: A new Adaptive Resampling Algorithm for Improving ANN Classifiers, Advances in Neural Information Processing Systems, 9, MIT press.

Quinlan, J.R.[1996] Bagging, Boosting, and C4.5, Proceedings of AAAI'96 National Conference on Artificial Intelligence, pp. 725-730.

Robinson, J.[1951] An Iterative Method of Solving a Game, Ann. Math 154, pp. 296-301

Schapire, R.,Freund, Y., Bartlett, P., and Lee, W[1997] Boosting the Margin, (available at http://www.research.att.com/~yoav look under "publications".)

Szep, J. and Forgo, F.[1985] Introduction to the Theory of Games, D. Reidel Publishing Co.

Appendix A Convergence of Type I Arcing Algorithms

Theorem . *Let* $\mathbf{b}^{(k)}$ *be the successive values generated by the Type I arcing algorithm, and set* $\mathbf{c}^{(k)}=\mathbf{b}^{(k)}/|\mathbf{b}|$ *Then* $\limsup_k top(\mathbf{c}^{(k)})\leq\phi$

proof. Clearly, $g(\mathbf{b}^{(k)})$ is decreasing in k. It suffices to show that $|\mathbf{b}^{(k)}|\rightarrow\infty$ since writing

$$er(\mathbf{z}_n,\mathbf{b}^{(k)})-\phi|\mathbf{b}^{(k)}|=|\mathbf{b}^{(k)}|(er(\mathbf{z}_n,\mathbf{c}^{(k)})-\phi)$$

shows that if there is a subsequence k' such that along this subsequence $top(\mathbf{c}^{(k')})\rightarrow\phi_1>\phi$, then $g(\mathbf{b}^{(k')})\rightarrow\infty$. If $|\mathbf{b}^{(k)}|$ does not go to infinity, then there is at least one finite limit point $\mathbf{b}^*$. But every time that $\mathbf{b}^{(k)}$ is in the vicinity of $\mathbf{b}^*$, $g(\mathbf{b}^{(k)})$ decreases in the next step by at least a fixed amount $\delta>0$. Since $g(\mathbf{b}^{(k)})$ is non-negative, this is not possible.

comments: i) From this argument, it's clear that cruder algorithms would also give convergence, since all that is needed is to generate a sequence $|\mathbf{b}^{(k)}|\rightarrow\infty$ such that $g(\mathbf{b}^{(k)})$ stays bounded. In particular, the line search can probably be avoided. ii) if we take (w.l.o.g) g(0)=1, then at the kth stage

$$|\{n: er(\mathbf{z}_n,\mathbf{c}^{(k)}) > \phi\} \leq Ng(\mathbf{b}^{(k)})|.$$

Appendix B Convergence of Type II Arcing Algorithms

Theorem *Let* $\mathbf{c}$ *be any stopping or limit point of the Type II arcing algorithm. Then* $\mathbf{c}$ *is a global minimum of* $g(\mathbf{c})$.

proof: Suppose there is a ϕ, $0<\phi<1$ such that $f'(x)>0$ for $x>\phi$ and zero for $x\le\phi$. If $\phi<\phi^*$ or if $f'(x)>0$ for all x then $\sum_n f'(er(\mathbf{z}_n,\mathbf{c}))=0$ is not possible. We treat this case first. Suppose the algorithm stops after a finite number of steps because $Q(e_{m*})\ge E_Q(er(\mathbf{z},\mathbf{c}))$. Then

$$\sum_n l_{m*}(\mathbf{z}_n)f'(er(\mathbf{z}_n,\mathbf{c}))\ge\sum_m c_m\sum_n l_m(\mathbf{z}_n)f'(er(\mathbf{z}_n,\mathbf{c})) \qquad \text{(B.1)}$$

This implies that for all m, either $c_m=0$ or

$$\sum_n l_{m*}(\mathbf{z}_n)f'(er(\mathbf{z}_n,\mathbf{c}))=\sum_n l_m(\mathbf{z}_n)f'(er(\mathbf{z}_n,\mathbf{c})) \qquad \text{(B.2)}$$

Consider the problem of minimizing $g(\mathbf{c})$under non-negativity and sum one constraints on $\mathbf{c}$. The Kuhn-Tucker necessary conditions are that there exist numbers λ and $\mu_m\ge0$ such that if $c_m>0$, then $\partial g(\mathbf{c})/\partial c_m=\lambda$. If $c_m=0$, then $\partial g(\mathbf{c})/\partial c_m=\lambda+\mu_m$. These conditions follow from (B.1) and (B.2). Because g($\mathbf{c}$) is convex in $\mathbf{c}$ these conditions are also sufficient.

Now suppose that the algorithm does not stop after a finite number of steps. After the kth step, let $\mathbf{c}^{(k+1)}$ be the updated $\mathbf{c}^{(k)}$ and m_k the index of the minimizing classifier at the kth step. Then

$$er(\mathbf{z}_n,\mathbf{c}^{(k+1)})-er(\mathbf{z}_n,\mathbf{c}^{(k)})=(l_{m_k}(\mathbf{z}_n)-er(\mathbf{z}_n,\mathbf{c}^{(k)}))/(k+1) \qquad \text{(B.3)}$$

Denote the right hand side of (B.3) by $\delta_k(\mathbf{z}_n)/(k+1)$. Using a partial Taylor expansion gives

$$g(\mathbf{c}^{(k+1)})-g(\mathbf{c}^{(k)})=\frac{1}{(k+1)}\sum_n\delta_k(\mathbf{z}_n)(f'(er(\mathbf{z}_n,\mathbf{c}^{(k)})+\frac{\gamma}{(k+1)^2} \qquad \text{(B.4)}$$

The first term on the right in (B.4) is negative for all k. Since $g(\mathbf{c})$ is bounded below for all $\mathbf{c}$,

$$\Sigma_k \frac{1}{(k+1)} |\Sigma_n \delta_k(\mathbf{z}_n)(f'(er(\mathbf{z}_n,\mathbf{c}^{(k)})| < \infty \qquad \text{(B.5)}$$

So, except possibly on a non-dense subsequence of the {k},

$$\Sigma_n \delta_k(\mathbf{z}_n)(f'(er(\mathbf{z}_n,\mathbf{c}^{(k)})) \to 0 \qquad \text{(B.6)}$$

Take a subsequence of the k for which (B.6) holds such that $m_k \to m^*$, $\mathbf{c}^{(k)} \to \mathbf{c}$. Then the situation of (B.2) is in force and **c** is a global minimum. Furthermore, since the first term on the right of (B.4) is negative (non-stopping), then (B.4) implies that the entire sequence $g(\mathbf{c}^{(k)})$ converges. Thus, all limits or stopping points of the $\mathbf{c}^{(k)}$sequence are global minimum points of $g(\mathbf{c})$.

Now examine the case $\phi \geq \phi^*$. If there is stopping because $\Sigma_n f'(er(\mathbf{z}_n,\mathbf{c}))=0$ then $top(\mathbf{c}) \leq \phi$ and $g(\mathbf{c})=Nf(0)$. Otherwise, note that for any **c**

$$\Sigma_n er(\mathbf{z}_n,\mathbf{c}) f'(er(\mathbf{z}_n,\mathbf{c})) \geq \phi \Sigma_n f'(er(\mathbf{z}_n,\mathbf{c})).$$

Hence

$$\Sigma_n (l_{m^*}(\mathbf{z}_n) - er(\mathbf{z}_n,\mathbf{c})) f'(er(\mathbf{z}_n,\mathbf{c})) \leq (\phi^* - \phi) \Sigma_n f'(er(\mathbf{z}_n,\mathbf{c})) \qquad \text{(B.7)}$$

If $\phi > \phi^*$ the right side of (B.7) is strictly negative and the algorithm never stops. Then (B.4) gives a subseqence satisfying (B.6). For any limit point **c** and m*,

$$\Sigma_n (l_{m^*}(\mathbf{z}_n) - er(\mathbf{z}_n,\mathbf{c})) f'(er(\mathbf{z}_n,\mathbf{c}))=0 \qquad \text{(B.8)}$$

which implies $top(\mathbf{c}) \leq \phi$ and $g(\mathbf{c})$. If $\phi = \phi^*$ and the algorithm stops, then (B.8) holds implying $top(\mathbf{c}) \leq \phi$. If it does not stop, the same conclusion is reached. In either case, we get $g(\mathbf{c})=Nf(0)$.

Appendix C Convergence of Arc-gv

Theorem *If arc-gv stops at the kth step, then* $top(\mathbf{c}^{(k)})=\phi^*$. *If it does not stop at any finite step, then* $\lim_k top(\mathbf{c}^{(k)})=\phi^*$.

proof. For an M-vector of weights **b** with $\mathbf{c}=\mathbf{b}/|\mathbf{b}|$ and $t=top(\mathbf{c})$, define

$$g(\mathbf{b})=\sum_n \exp(er(\mathbf{z}_n,\mathbf{b})-t|\mathbf{b}|)$$

$$Q_{\mathbf{b}}(\mathbf{z}_n)=\exp(er(\mathbf{z}_n,\mathbf{b})-t|\mathbf{b}|)/S$$

Consider increasing the mth coordinate of **b** by the amount Δ to get **b'.** Let

$$\Theta(m,\mathbf{b},\Delta)=E_{Q_{\mathbf{b}}}(\exp(\Delta(l_m(\mathbf{z})-t).$$

Then this identity holds:

$$g(\mathbf{b}')=\Theta(m,\mathbf{b},\Delta)g(\mathbf{b})\exp((|\mathbf{b}'|(t-t'))) \tag{C.1}$$

where $t'=top(\mathbf{c}')$.

<u>**Proposition**</u> *If* $t-E_{Q_{\mathbf{b}}}l_m=\mu_{\mathbf{b}}>0$ *then*

$$\min_{\Delta}\Theta(m,\mathbf{b},\Delta)\leq 1-.5\mu_{\mathbf{b}}{}^2$$

where the minimum is over the range $[0,1]$.

<u>proof.</u> Abbreviate $\Theta(m,\mathbf{b},\Delta)$by $\Theta(\Delta)$. Using a partial expansion gives

$$\Theta(\Delta)=1-\mu_{\mathbf{b}}\Delta+(\Delta^2/2)\Theta''(\alpha\Delta), \quad 0\leq\alpha\leq 1$$

Now,

$$\Theta''(\alpha\Delta)=E_{Q_{\mathbf{b}}}[(l_m-t)^2\exp(\alpha\Delta((l_m-t))]\leq\Theta(\alpha\Delta)$$

Let [0,s] be the largest interval on which $\Theta(\Delta)\leq 1$. On this interval

$$\Theta(\Delta)\leq 1-\mu_{\mathbf{b}}\Delta+\Delta^2/2 \tag{C.2}$$

The right hand side of (C.2) has a minimum at $\Delta^*=\mu_{\mathbf{b}}$ and

$$\Theta(\Delta^*)\leq 1-\mu_{\mathbf{b}}^2/2 \tag{C.3}$$

Note that $\Delta^* \leq 1$ is in the [0,1] range.

To analyze the behavior of arc-gv we introduce the following notation

$\mathbf{b}^{(k)}$: the vector of weights after the kth step,

E_k : the expectation w.r.t. to $Q_{\mathbf{b}^{(k)}}$,

Θ_k: the minimum of $\Theta(m_{k+1}, \mathbf{b}^{(k)}, \Delta)$ over the interval $[0,\bar{\Delta}]$,
Δ_k: the minimizing value of Δ.

and set $\mu_k = \mu_{\mathbf{b}^{(k)}}$, $g_k = g(\mathbf{b}^{(k)})$. By (C.1)

$$\log(g_{k+1}) = \log(g_k) + |\mathbf{b}^{(k+1)}|(t_k - t_{k+1}) + \log\Theta_k \qquad \text{(C.4)}$$

Summing (C.4) gives

$$\log(g_{k+1}) = \log(N) + \sum_{j=1}^{k}[|\mathbf{b}^{(j+1)}|(t_j - t_{j+1}) + \log\Theta_j] \qquad \text{(C.5)}$$

Rearranging the sum on the right of (C.5) gives

$$\log(g_{k+1}) = \log(N) + \sum_{j=1}^{k}[\Delta_j(t_j - t_{k+1}) + \log\Theta_j]$$

For any $\mathbf{b}$, since $\min_m E_{Q_{\mathbf{b}}} l_m \leq \phi^*$ then $\min_m E_{Q_{\mathbf{b}}} l_m \leq top(\mathbf{c})$ with equality only if $top(\mathbf{c}) = \phi^*$. Now $\mu_k = t_k - \min_m E_k l_m$ so $\mu_k \geq 0$ only if $t_k = \phi^*$. But this is just the stopping condition. If there is no stopping then all $\mu_j > 0$ and

$$\log(g_{k+1}) \leq \log(N) + \sum_{j=1}^{k}[\Delta_j(t_j - t_{k+1}) - \mu_j^2/2] \qquad \text{(C.6)}$$

Since $\log(g_{k+1}) \geq 0$ the sum on the right of (C.6) must be bounded below.

Take a subsequence $\{k'\}$ such that $t_{k'+1} \to \limsup t_k = \bar{t}$ and look at (C.6) along this subsequence assuming $\bar{t} = \phi^* + \delta$ where $\delta > 0$. Let $N_{k'}$ be the number of terms in the sum in (C.6) that are positive. We claim that $\sup_{k'} N_{k'} < \infty$. To show this, suppose the jth term is positive, i.e.

$$t_j > t_{k'+1} + \mu_j^2/2 \tag{C.7}$$

If $t_j \geq \phi^* + \tau,\ \tau > 0$ then $\mu_j \geq \tau$. This implies that for k' sufficiently large, there is a fixed $\varepsilon > 0$ such that if (C.7) is satisfied, then $t_j \geq \bar{t} + \varepsilon$. But this can happen at most a finite number of times.

Let the sum of the positive terms in (C.6) plus log(N) be bounded by S. Fix $\varepsilon > 0$. In the negative terms in the sum, let j' index those for which $|t_j - t_{k'+1}| \leq \varepsilon$. Then

$$\log(g_{k'+1}) \leq S + \sum_{j'} (\varepsilon - \mu_{j'}^2/2) \tag{C.8}$$

Take $\varepsilon \leq \delta/2$. For k' large enough and all j', $t_{j'} > \phi^* + \delta/2$ and $\mu_{j'}^2 \geq \delta^2/4$. Taking ε so that $\varepsilon < \delta^2/16$ shows that the number of terms in the (C.6) sum such that $|t_j - t_{k'+1}| \leq \varepsilon$ is uniformly bounded. This contradicts that fact that the $t_{k'+1}$ sequence converges to a limit point unless $\limsup t_k = \phi^*$.

<u>Finding the minimizing Δ.</u>

The minimizing Δ is given by a simple expression. By its definition,
.

$$\Theta(m, \mathbf{b}, \Delta) = e^{-\Delta t}[1 + (e^{\Delta} - 1)Q_{\mathbf{b}}(e_m)].$$

Setting the derivation of Θ with respect to Δ equal to zero and solving gives

$$\Delta = \log\left[\frac{t}{1-t}\frac{1-q}{q}\right]$$

where $q = Q_{\mathbf{b}}(e_m)$.

<u>Appendix D Upper Bound for the Generalization Error in Terms of Top(c)</u>

<u>**Theorem**</u> *For* $\Delta > 0$ *, define*

$$R=\frac{8\log(2M)}{N\Delta^2}$$

Except for a set of training sets with $\tilde{P}$ *probability* $\le\delta$, *for every* $\Delta\ge\sqrt{8/M}$ *and* $\mathbf{c}$

$$P(er(\mathbf{Z},\mathbf{c})\ge\Delta+top(\mathbf{c}))\le R(1+\log(1/R)+\log(2N))+(\log(M)/\delta)/N$$

<u>Proof</u> Denote $l(m,\mathbf{z})=I(\mathbf{z}\in e_m)$ where I is the indicator function. Let K to be a positive integer and fixing $\mathbf{c}$ take $J_k^*, k=1,\dots,K$ to be independent random variables such that $P(J_k^*=m)=c_m$. Denote by $\mathbf{J^*}$ the random K-vector whose kth component is J_k^*. Conditional on $\mathbf{Z}$

$$\bar{L}(\mathbf{Z},\mathbf{J}^*)=\frac{1}{K}\sum_1^K l(J_k^*,\mathbf{Z})$$

is an average of iid random variables, each one having expectation $er(\mathbf{Z},\mathbf{c})$. Similarly,

$$\bar{L}(\mathbf{z}_n,\mathbf{J}^*)=\frac{1}{K}\sum_1^K l(J_k^*,\mathbf{z}_n)$$

is an average of iid variables, each having expectation $er(\mathbf{z}_n,\mathbf{c})$. For any $\mu<\lambda$

$$P(er(\mathbf{Z},\mathbf{c})\ge\lambda)\le P(\bar{L}(\mathbf{Z},\mathbf{J}^*)\ge\mu)+P(\bar{L}(\mathbf{Z},\mathbf{J}^*)<\mu,er(\mathbf{Z},\mathbf{c})\ge\lambda) \qquad \text{(D.1)}$$

Bound the 2nd term on the right of (D.1) by

$$\begin{aligned}&E(P(\bar{L}(\mathbf{Z},\mathbf{J}^*)<\mu,er(\mathbf{Z},\mathbf{c})\ge\lambda|\mathbf{Z}))\\ &\le E(P(\bar{L}(\mathbf{Z},\mathbf{J}^*)-E(\bar{L}(\mathbf{Z},\mathbf{J}^*)|\mathbf{Z})<\mu-\lambda|\mathbf{Z}))\end{aligned} \qquad \text{(D.2)}$$

By a version of the Chernoff inequality, the term on the right of (D.2) is bounded by

$$\exp((-K(\mu-\lambda)^2/2) \qquad \text{(D.3)}$$

To bound the 1st term in (D.1), for some $\varepsilon>0$ consider the probability

$$\tilde{P}(E\{P(\bar{L}(\mathbf{Z},\mathbf{J}^*)\geq\mu|\mathbf{J}^*)-\max_n I(\bar{L}(\mathbf{z}_n,\mathbf{J}^*)\geq\mu)\}\geq\varepsilon) \tag{D.4}$$

where $\tilde{P}$ is the probability measure on the sets of N-instance training sets T. Let **j** denote any of the values of **J*.** (D.4) is bounded above by

$$\tilde{P}(\max_{\mathbf{j}}\{P(\bar{L}(\mathbf{Z},\mathbf{j})\geq\mu)-\max_n I(\bar{L}(\mathbf{z}_n,\mathbf{j})\geq\mu)\}\geq\varepsilon)$$
$$\leq\sum_{\mathbf{j}}\tilde{P}(\{P(\bar{L}(\mathbf{Z},\mathbf{j})\geq\mu)-\max_n I(\bar{L}(\mathbf{z}_n,\mathbf{j})\geq\mu)\}\geq\varepsilon) \tag{D.5}$$

By the independence of the $\{\mathbf{z}_n\}$ under $\tilde{P}$, the **j**th term in (D.5) is bounded by

$$\exp(-N(P\{P(\bar{L}(\mathbf{Z},\mathbf{j})\geq\mu)-I(\bar{L}(\mathbf{Z},\mathbf{j})\geq\mu)\}\leq\varepsilon)) \tag{D.6}$$

We can lower bound the term multiplied by $-N$ in the exponent of (D.6).

$$P\{P(\bar{L}(\mathbf{Z},\mathbf{j})\geq\mu)-I(\bar{L}(\mathbf{Z},\mathbf{j})\geq\mu)\}\leq\varepsilon)=$$
$$P\{P(\bar{L}(\mathbf{Z},\mathbf{j})\geq\mu)\leq\varepsilon+1,\bar{L}(\mathbf{Z},\mathbf{j})\geq\mu\}+P\{P(\bar{L}(\mathbf{Z},\mathbf{j})\geq\mu)\leq\varepsilon,\bar{L}(\mathbf{Z},\mathbf{j})<\mu\}=$$
$$P(\bar{L}(\mathbf{Z},\mathbf{j})\geq\mu)+I(P(\bar{L}(\mathbf{Z},\mathbf{j})\geq\mu)\leq\varepsilon)P(\bar{L}(\mathbf{Z},\mathbf{j})<\mu)\geq\varepsilon$$

Hence, (D.4) is bounded by $M^k\exp(-\varepsilon N)$. Take a grid of M values $\mu_i, i=1,\ldots,M$ equispaced in [0,1]. Then the probability

$$\tilde{P}(\max_i E\{P(\bar{L}(\mathbf{Z},\mathbf{J}^*)\geq\mu_i|\mathbf{J}^*)-\max_n I(\bar{L}(\mathbf{z}_n,\mathbf{J}^*)\geq\mu_i)\}\geq\varepsilon)$$

bounded by $M^{k+1}\exp(-\varepsilon N)$. To bound another term, take $\nu<\mu$ and write

$$E(\max_n I(\bar{L}(\mathbf{z}_n,\mathbf{J}^*)\geq\mu))=P(\max_n \bar{L}(\mathbf{z}_n,\mathbf{J}^*)\geq\mu)\leq$$

$$I(\max_n er(\mathbf{z}_n,\mathbf{c})>\nu)+P(\max_n \bar{L}(\mathbf{z}_n,\mathbf{J}^*)\geq\mu,\ \max_n er(\mathbf{z}_n,\mathbf{c})\leq\nu) \tag{D.7}$$

The last term in (D.7) is bounded by

$$P(\max_n(\bar{L}(\mathbf{z}_n,\mathbf{J}^*)-er(\mathbf{z}_n,\mathbf{c}))\geq\mu-\nu)\leq N\exp(-K(\mu-\nu)^2/2) \tag{D.8}$$

For any λ,ν take μ to be the lowest value in the grid of M μ-values that is $\geq(\lambda+\nu)/2$. So

$$\mu = \frac{(\lambda+\nu)}{2} + \frac{\alpha}{M}$$

where $0 \le \alpha < 1$. Assuming that $0 < \lambda - \nu \le 1$ the sum of the bounds in (D.3) and (D.8) is less than

$$S_K = \max(2N, \exp(K/2M)) \exp(-K(\lambda-\nu)^2/8).$$

Let the ε in (D.4) depend on K. and define $\delta_K = M^{K+1} \exp(-\varepsilon_K N)$. Then, except for a fixed set of training sets with $\tilde{P}$ probability $\le \delta_K$, for all $\lambda, \nu, \mathbf{c}$, and for fixed K,

$$P(er(\mathbf{Z},\mathbf{c}) \ge \lambda) \le \varepsilon_K + S_K + I(top(\mathbf{c}) > \nu) \tag{D.9}$$

Take $\delta_K = 2^{-K}\delta$. Then (D.9) also holds for all K except on a fixed set of training sets with probability$\le \delta$. Now let $\nu = top(\mathbf{c})$, $\lambda = \Delta + top(\mathbf{c})$, $\sigma = 8/\Delta^2$, and take $K = \sigma \log(2N^2/\sigma \log(2M))$. If $\Delta \ge \sqrt{8/M}$, then $2N \ge \exp(K/2M)$ and letting $R = \sigma \log(2M)/N$ gives:

$$P(er(\mathbf{Z},\mathbf{c}) \ge \Delta + top(\mathbf{c})) \le R(1 - \log R + \log(2M)) + (\log(2M/\delta))/N$$

which is the assertion of the theorem.

FIGURE 1

CUMULATIVE MARGIN DISTRIBUTIONS

ADABOOST ooooo ARC-GV oooooooo

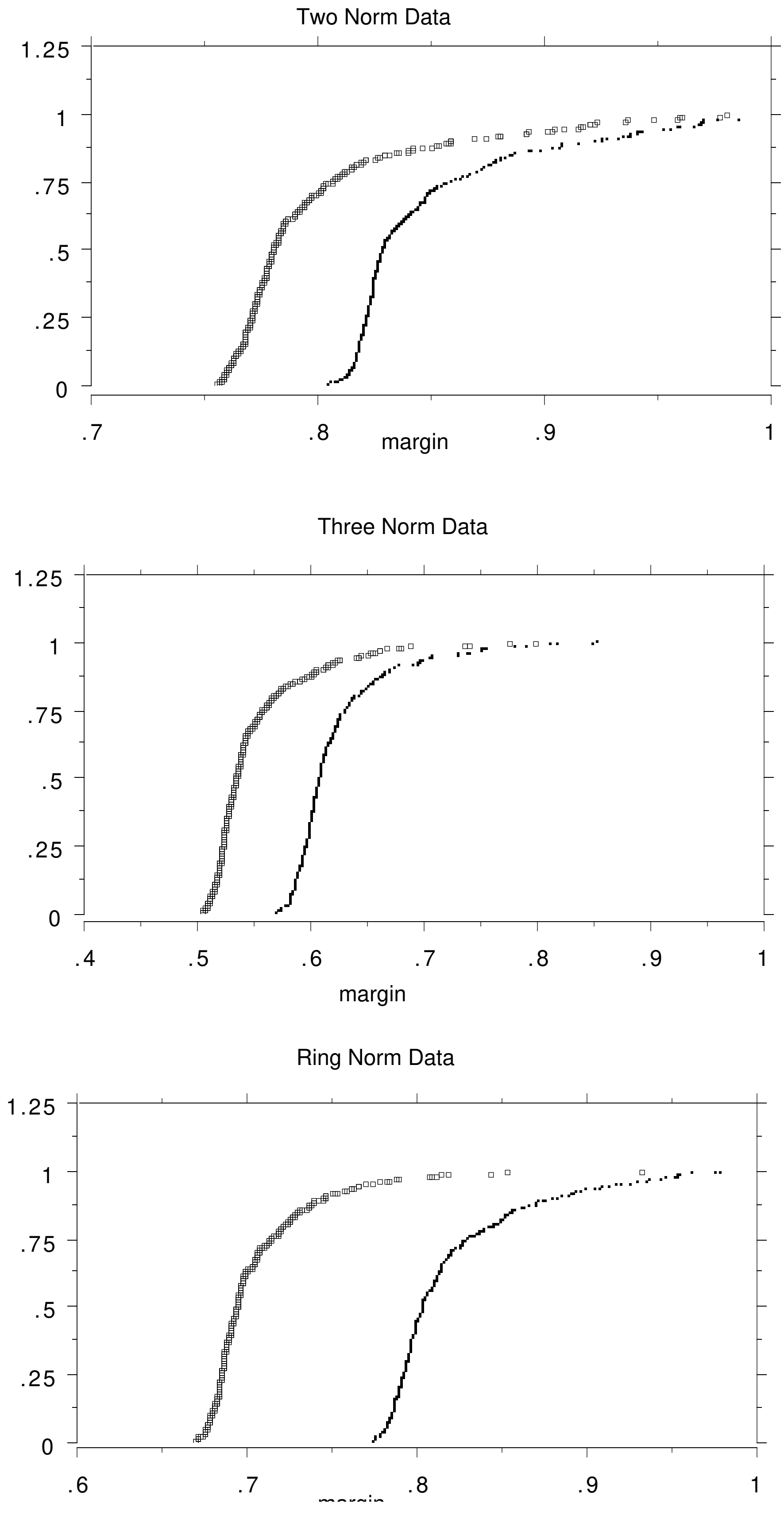

FIGURE 2

CUMULATIVE MARGIN DISTRIBUTIONS

ADABOOST ooooo ARC-GV oooooooo