PREDICTION GAMES AND ARCING ALGORITHMS

Leo Breiman
leo@stat.berkeley.edu

Technical Report 504
December 19, 1997
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].[1996]) and others in reducing generalization error has not been well understood. By formulating prediction, both classification and regression, 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. An optimal game strategy finds a combined predictor that minimizes the maximum of the error over the training set. 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. Arcing algorithms are described that converge to the optimal strategy. Schapire et.al. [1997] offered an explanation of why Adaboost works in terms of its ability to reduce the margin. Comparing Adaboost to our optimal arcing algorithm shows that their explanation is not valid and that the answer lies elsewhere. In this situation the VC-type bounds are misleading. Some empirical results are given to explore the situation.

1. **Introduction**

Recent work empirical work has shown that combining predictors can lead to significant reduction in generalization error (Breiman[1996].1997]. Drucker and Cortes[1995], Quinlan [1996], Freund and Schapire[1996a], Kong and Dietterich[1996], Bauer and Kohavi[1998]). Interestingly, the individual predictors can be very simple, i.e. single hyperplanes in two class classification (Ji and Ma[1997]) .

Let $\{h_m(\mathbf{x})\}$ be a set of M predictors defined on input vectors $\mathbf{x}$. In regression $h_m(\mathbf{x})$ is a numerical value--in classification, 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 regressions predict the weighted average

$$\sum c_m h_m(\mathbf{x}).$$

The combined classifications 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* $\{h_m(\boldsymbol{x})\}$ *find* $\{c_m\}$ *such that the combined predictor has low generalization error.*

Given an input-output pair $\mathbf{z}=(y,\mathbf{x})$, in regression, define the squared error of a combined predictor as:

$$rs(\mathbf{z},\mathbf{c})=(y-\sum c_m h_m(\mathbf{x}))^2.$$

In classification, define the margin error at $\mathbf{z}$ by

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

If the margin error at $\mathbf{z}$ is negative, the right class is predicted. The more negative it is, the larger "the margin of correctness".

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

$E(rs(\mathbf{Z},\mathbf{c}))$ regression
$P(mg(\mathbf{Z},\mathbf{c})>0)$ classification.

2. **The Prediction Games**

Let $er(\mathbf{z},\mathbf{c})$ equal $rs(\mathbf{z},\mathbf{c})$ in regression and $mg(\mathbf{z},\mathbf{c})$ in classification. 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 see that these two questions are linked.

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

In both regression and classification $er(\mathbf{z}_n,\mathbf{c})$ is continuous and convex 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. Noting that

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

lets us rewrite (2.1) as

$$\phi^*=\inf_{\mathbf{c}} top(\mathbf{c})=\sup_Q\inf_{\mathbf{c}} E_Q er(\mathbf{z},\mathbf{c}) \qquad (2.2)$$

3. **The Dominating Matrix Game**

In both regression and classification, $er(\mathbf{z},\mathbf{c})$is dominated by a linear function of $\mathbf{c}$.

Regression:

$$er(\mathbf{z},\mathbf{c})\leq\sum c_m(y-h_m(\mathbf{x}))^2$$

Classification:

$$er(\mathbf{z},\mathbf{c})\leq 2\sum c_m I(y\neq h_m(\mathbf{x}))-1 \qquad (3.1)$$

If there are only two classes then (3.1) becomes an equality.

In both cases the dominating function is of the form:

$$er_M(\mathbf{z},\mathbf{c})=\sum c_m l(y,h_m(\mathbf{x}))$$

where $l(y,h_m(\mathbf{x}))$ is a non-negative loss function and the subscript M refers to the matrix game. Choosing $\mathbf{c}$ to make $er_M(\mathbf{z},\mathbf{c})$ small on T will force $er(\mathbf{z},\mathbf{c})$ to be small. In particular: for regression

$$top(\mathbf{c})\leq top_M(\mathbf{c})$$

and in classification

$$top(\mathbf{c})\leq 2top_M(\mathbf{c})-1 \qquad (3.2)$$

with equality if there are only two classes. To simplify notation, denote $l_m(\mathbf{z})=l(y,h_m(\mathbf{x}))$.

Definition 3.1 *The dominating matrix game is a two person zero-sum game. Player I chooses* $\mathbf{z}_n\varepsilon T$. *Player II chooses m. Player I wins the amount* $l_m(\mathbf{z}_n)$.

In this simpler game, there is a good mixed strategy $\{c_m\}$ for Player II and a good mixed strategy for Player I consisting of a probability measure over the instances in T. The value of the game ϕ^* is given by:

$$\phi^*=\inf_{\mathbf{c}} top_M(\mathbf{c})=\sup_Q \min_m E_Q l_m(\mathbf{z}) \qquad (3.3)$$

The relations (2.2) and (3.3) will be the keys in our construction of effective algorithms to find optimum **c** values. The definition of the matrix game for classification appeared in Freund and Schapire[1996b].

The relation (3.3) also follows from putting the game into a linear programming context. Define ϕ^* to be the minimum value of top(**c**). Consider minimizing the function $\sum_n (er_M(\mathbf{z}_n,\mathbf{c})-\phi)^+$ where x^+ is the positive part of x. If $\phi>\phi^*$ then the minimum is zero. Denoting $p_n=Q(\mathbf{x}_n)$, the minimization can also be cast as the linear programming problem: minimize $\sum_n u_n p_n$ under the constraints

$$u_n \geq 0, u_n \geq \sum_m c_m l_m(\mathbf{z}_n)-\phi, c_m \geq 0, \sum_m c_m = 1$$

The dual problem is: maximize $-\phi\sum_n \lambda_n + z$ under the constraints

$$\lambda_n \geq 0, \lambda_n \leq p_n, z \leq \sum_n \lambda_n l_m(\mathbf{z}_n)$$

This is equivalent to: maximize $-\phi\sum_n \lambda_n + \min_m \sum_n \lambda_n l_m(\mathbf{z}_n)$ under the constraints $0\leq\lambda_n\leq p_n$. Suppose there is a set of $\{\lambda_n\}$ that make this expression positive. Setting $Q(\mathbf{x}_n)=\lambda_n/\sum_n \lambda_n$, then $\min_m Q(e_m)>\phi$ implying $\phi \geq \phi^*$. On the other hand, take Q so that $\varphi=\min_m Q(e_m)$ and let $\lambda_n = rQ(\mathbf{x}_n)$ where r >0 is taken so that $\lambda_n \leq p_n$, all n. Then the maximum is bounded below by $r(\varphi-\phi)$, so $\varphi\leq\phi^*$. Thus, (3.3) follows.

4. A Bound on the Generalization Error

Schapire et.al[1997] derived a bound on the classification generalization error in terms of the distribution of the distribution of $mg(\mathbf{z},\mathbf{c})$ on the instances of T. Using the same elegant device that they created, we can derive a sharper bound using the value of $top_M(\mathbf{c})$ 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 $er_M(\mathbf{Z},\mathbf{c})=\sum_m c_m l_m(\mathbf{Z})$. Denote by $\tilde{P}$ the probability on the set of all N-instance training sets such that each one is drawn from the distribution P. Assume there is a constant B such that for all m, $0\leq l_m(\mathbf{Z})\leq B$. Set $\delta>0$. Then:

Theorem 4.1 *For* $\Delta > 0$ *, define*

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

Except for a set of training sets with $\tilde{P}$ *probability* $\leq \delta$*, for every* Δ *and* $\mathbf{c}$

$$P(er_M(\mathbf{Z},\mathbf{c}) \geq \Delta + top_M(\mathbf{c})) \leq R(1+\log(1/R)+\log(2N))+(\log(M)/\delta)/N \qquad (4.1)$$

The proof of this theorem is given in the Appendix. The bound in Schapire et.al.[1997] is about square root of the bound in (4.1). The additional .sharpness comes from using the uniform bound given by $top_M(\mathbf{c})$. The bound gives non-trivial results only if R is small. We give this theorem and its proof mainly as a comfort 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])

5. Arcing Algorithms for the Matrix Game.

For the matrix game, the main problem is how to determine $\mathbf{c}$ so that $er_M(\mathbf{z},\mathbf{c})$ 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, a linear programming method can be used to find a $\mathbf{c}$ such that $\phi^* = top_M(\mathbf{c})$.

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 more than one at a time as required by a standard optimization method. 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* $E_Q l(y,h_m(\mathbf{x}))$

This means, in regression, find that predictor having the lowest Q-weighted least squares error. In classification, minimize the Q-weighted missclassification 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 5.1 *An arcing algorithm for the matrix game 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* $E_Q l(y,h_m(\mathbf{x}))$
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**ewighting and **C**ombin**i n g** (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 sections 6,7,8 we will give examples of arcing algorithms in the matrix classification game, together with gereral descriptions and convergence properties. To simplify notation we will drop the subscript M indicating matrix game and write $\text{top}(\mathbf{c})=top_M(\mathbf{c})$ and $er(\mathbf{z},\mathbf{c})=er_M(\mathbf{z},\mathbf{c})$.

6. Examples of Arcing Algorithms,

There are a number of successful arcing algorithms appearing in recent literature. We give three examples in classification 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

$$\varepsilon_k=\sum_n l_m(\mathbf{z}_n)Q_k(\mathbf{z}_n)$$

Set $\beta_k=(1-\varepsilon_k)/\varepsilon_k$, and

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

where the /S indicates normalization to sum one. The increase in b_m is $\log(\beta_k)$.

Example 2. Arc-x4 (Breiman [1997])

Define $ms_{(k)}(\mathbf{z}_n)$as the number of missclassifications 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, increase b_m by one.

Example 3. Random Hyperplanes (Ji and Ma[1997])

This method applies only to two class problems. The set of classifiers H is defined this way: To each direction in input space and point $\mathbf{x}_n$ in the training set corresponds two classifiers. Form the hyperplane passing through $\mathbf{x}_n$ perpendicular to the given direction. 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, increase its b weight by one. Let

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

where I(·) 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..

The existing arcing algorithms (see also Leisch and Hornik[1997]) fall into two different types which will be defined and discussed in the next two sections. In these sections, we let $e_m=\{\mathbf{z}_n; y_n\neq h_m(\mathbf{x}_n)\}$, i.e. e_m is the set of instances for which $l_m(\mathbf{z}_n)=1$..

7 Unnormalized Arcing Algorithms

Let f(x) $\to\infty$ as $x\to\infty$, to 0 as $x\to-\infty$, and have everywhere positive first and second derivatives. For weights $\{b_m\}$ we slightly abuse notation and set $er(\mathbf{z}_n,\mathbf{b})=\sum_m b_m l_m(\mathbf{z}_n)$. Assuming that $\phi>\phi^*$, we want to minimize $g(\mathbf{b})=\sum_n f(er(\mathbf{z}_n,\mathbf{b})-\phi|\mathbf{b}|)$ starting from $\mathbf{b}=0$.

Definition 7.1 *The unnormalized arcing algorithm updates* $\mathbf{b}$ *as follows: At the current value of* $\mathbf{b}$ *let*

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

and $m^*=argmin_m\, Q(e_m)$. *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* $\mathbf{b}\rightarrow\mathbf{b}+\Delta^*\,\mathbf{u}_{m^*}$. *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, $\min_m E_Q(e_m)\leq\phi^*$. 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 7.2. *Let* $\mathbf{b}^{(k)}$ *be the successive values generated by the unnormalized arc 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 **b*.** But every time that $\mathbf{b}^{(k)}$ is in the vicinity of **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, its clear that cruder algorithms would also give convergence, since all that is 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 be avoided (see Section 8). ii) if we take (w.l.o.g) g(0)=1, then at the kth stage $\#\{n;er(\mathbf{z}_n,\mathbf{c}^{(k)})>\phi\}\leq Ng(\mathbf{b}^{(k)})$.

Proposition 7.3. *Adaboost is an unnormalized 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 by $\pi(\mathbf{z}_n)\rightarrow\pi(\mathbf{z}_n)e^{\Delta^* l_m(\mathbf{z}_n)}$. For $\phi=1/2$ this is the Adaboost algorithm described in Section 1.

Schapire et.al.[1997] note that the Adaboost algorithm produces a $\mathbf{c}$ sequence so that limsup(top($\mathbf{c}$) is less than 1/2. In fact, we can show that limsup(top($\mathbf{c}$))$\le$(log2+log(1$-\phi$*))/(−log(ϕ*)+log(1−ϕ*)). If, for instance, ϕ*=.25, this bound equals .37. Thus, even though Theorem 7.2 only guarantees an upper bound of .5, using the exponential form of f allows a sharper bound to be given.

8 Normalized Arc Algorithms

The normalized 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$\in$ [0,1]. Let $\mathbf{c}=\mathbf{b}/|\mathbf{b}|$.

Definition 8.1 *The normalized arc algorithm updates* $\mathbf{b}$ *as follows: At the current value of* $\mathbf{b}$ *, if* $\sum_n f'(er(\mathbf{z}_n,\mathbf{c}))=0$ *then stop. Otherwise, let*

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

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

Comments: 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 8.2 *Let* $\mathbf{c}$ *be any stopping or limit point of the normalized arc 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 $x\leq\phi$. 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 $Q(e_{m^*})\geq E_Q(er(\mathbf{z},\mathbf{c}))$ Then

$$\sum_n l_{m^*}(\mathbf{z}_n)\, f'(er(\mathbf{z}_n,\mathbf{c}))\geq\sum_m c_m \sum_n l_m(\mathbf{z}_n)\, f'(er(\mathbf{z}_n,\mathbf{c})) \tag{8.1}$$

This implies that for all m, either $c_{m=0}$ or

$$\sum_n l_{m^*}(\mathbf{z}_n)\, f'(er(\mathbf{z}_n,\mathbf{c}))=\sum_n l_m(\mathbf{z}_n)\, f'(er(\mathbf{z}_n,\mathbf{c})) \tag{8.2}$$

Consider the problem of minimizing g(**c**) under non-negativity and sum one constraints on **c**. The Kuhn-Tucker necessary conditions are that there exist numbers λ and $\mu_m\geq 0$ 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 (8.1) and (8.2). Because g(**c**) is convex in **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

$$er(\mathbf{z}_n,\mathbf{c}^{(k+1)})-er(\mathbf{z}_n,\mathbf{c}^{(k)})=(l_{m_k}(\mathbf{z}_n)-er(\mathbf{z}_n,\mathbf{c}^{(k)}))/(k+1) \tag{8.3}$$

Denote the right hand side of (8.3) by $\delta_k(\mathbf{z}_n)/(k+1)$. Using a partial Taylor expansion gives

$$g(\mathbf{c}^{(k+1)})-g(\mathbf{c}^{(k)})=\frac{1}{(k+1)}\sum_n \delta_k(\mathbf{z}_n)(\, f'(er(\mathbf{z}_n,\mathbf{c}^{(k)})+\frac{\gamma}{(k+1)^2} \tag{8.4}$$

The first term on the right in (8.4) is negative for all k. Since g(**c**) is bounded below for all **c**,

$$\sum_k \frac{1}{(k+1)}|\sum_n \delta_k(\mathbf{z}_n)(\, f'(er(\mathbf{z}_n,\mathbf{c}^{(k)})|<\infty \tag{8.5}$$

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

$$\sum_n \delta_k(\mathbf{z}_n)(\, f'(er(\mathbf{z}_n,\mathbf{c}^{(k)}))\rightarrow 0 \tag{8.6}$$

Take a subsequence of the k for which (8.6) holds such that $m_k \to m^*, \mathbf{c}^{(k)} \to \mathbf{c}$. Then the situation of (8.2) is in force and **c** is a global minimum. Furthermore, since the first term on the right of (8.4) is negative (non-stopping), then (8.4) 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(**c**).

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

$$\sum_n (l_{m^*}(\mathbf{z}_n) - er(\mathbf{z}_n,\mathbf{c})) f'(er(\mathbf{z}_n,\mathbf{c})) \leq (\phi^* - \phi) \sum_n f'(er(\mathbf{z}_n,\mathbf{c})) \quad (8.7)$$

If $\phi > \phi^*$ the right side of (8.7) is strictly negative and the algorithm never stops. Then (8.4) gives a subseqence satisfying (8.6). For any limit point **c** and m*, $\sum_n (l_{m^*}(\mathbf{z}_n) - er(\mathbf{z}_n,\mathbf{c})) f'(er(\mathbf{z}_n,\mathbf{c}))=0$ which implies $top(\mathbf{c}) \leq \phi$ and g(**c**)=0. If $\phi=\phi^*$ and the algorithm stops, $\sum_n (l_{m^*}(\mathbf{z}_n) - er(\mathbf{z}_n,\mathbf{c})) f'(er(\mathbf{z}_n,\mathbf{c}))=0$, implying $top(\mathbf{c}) \leq \phi$. If it does not stop, the same conclusion is reached. In either case, we get g(**c**)=Nf(0).

One version of the normalized algorithm starts with a function q(x) defined on [-1,1] such that q'(x) is zero for x≤0, positive for x>0, and q" continuous, bounded and non-negative. For $\phi > \phi^*$ define f(q)=g(x-ϕ). Applying the algorithm to this f(x) gives the following result:

Corollary 8.3 *For* ***c*** *any limit or stopping point of the normalized algorithm,* $top(\mathbf{c}) \leq \phi$

proof: $top(\mathbf{c}) \leq \phi$ is necessary and sufficient for a global minimum of g(**c**).

Proposition 8.4 *Arc-x4 is a normalized arc algorithm*

Proof. In normalized 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 missclassifications 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 $\sum_n er(\mathbf{z}_n,\mathbf{c})^5$.

Proposition 8.5 *Random Hyperplanes is (almost) a normalized arc algorithm*

Proof. Take $\phi>\phi^*$and consider trying to minimize $\sum_n(er(\mathbf{z}_n,\mathbf{c})-\phi)^+$using the normalized algorithm. At each stage, the current probability $Q(\mathbf{z}_n)$ is proportional to $I(er(\mathbf{z}_n,\mathbf{c})-\phi>0)$ where $I(\bullet)$ is the indicator function, and this is the Ji-Ma reweighting. In the standard form of the normalized 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 normalized arc algorithm has to be modified a bit to make the convergence proof work.

Take $\varepsilon>0$ small, and q(x) 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 Theorem 8.2 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,β by an experimental search. This is not surprising since the value of $\phi^*$is unknown.

9. **An Optimizing Arcing Algorithm for the Matrix Game.**

This section describes an arcing algorithm we call arc-gv (gv=game value) and proves that $top(\mathbf{c}^{(k)})\to\phi^*$. Let B be such that for all m,n $0\le l_m(\mathbf{z}_n)\le B$ and set $\overline{\Delta}$ to be a number much larger than 1/B. The algorithm generates a sequence of weight vectors $\mathbf{b}^{(k)}$ and the normed weights are $\mathbf{c}^{(k)}=\mathbf{b}^{(k)}/|\mathbf{b}^{(k)}|$ 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 9.1 *Arc-gv updates* $\mathbf{b}^{(k)}$ *to* $\mathbf{b}^{(k+1)}$ *as follows:*

i) Let

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

$$m_{k+1}=\arg\min_m E_{Q_k} l_m$$

ii) Let Δ_k *be the minimizer of*

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

in the range $[0,\bar{\Delta}]$.

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

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

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

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:

$$g(\mathbf{b}')=\Theta(m,\mathbf{b},\Delta)g(\mathbf{b})\exp((|\mathbf{b}'|(t-t'))) \qquad (9.1)$$

where t'=top(**c'**).

Proposition 9.3 *If* $t-E_{Q_{\mathbf{b}}}l_m=\mu_{\mathbf{b}}>0$ *then*

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

where the minimum is over the range $[0,\bar{\Delta}]$.

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

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

Now,

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

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

$$\Theta(\Delta)\le 1-\mu_{\mathbf{b}}\Delta+(B^2\Delta^2/2) \tag{9.2}$$

The right hand side of (9.2) has a minimum at $\Delta^*=\mu_{\mathbf{b}}/B^2$ and at Δ^*

$$\Theta(\Delta^*)\le 1-\mu_{\mathbf{b}}^2/2B^2 \tag{9.3}$$

Note $\Delta^*\le 1/B\le\bar{\Delta}$ so Δ^* is in the $[0,\bar{\Delta}]$ range.

To analyze the the behavior of arc-gv we use the notation that $\mathbf{b}^{(k)}$ is the vector of weights after the kth step, E_k the expection w.r. to $Q_{\mathbf{b}^{(k)}}$, $t_k=top(\mathbf{c}^{(k)})$, $\mu_k=\mu_{\mathbf{b}^{(k)}}$, 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 $g_k=g(\mathbf{b}^{(k)})$. By (9.1)

$$\log(g_{k+1})=\log(g_k)+|\mathbf{b}^{(k+1)}|(t_k-t_{k+1})+\log\Theta_k \tag{9.4}$$

Summing (9.4) gives

$$\log(g_{k+1})=\log(N)+\sum_{j=1}^{k}|\mathbf{b}^{(j+1)}|(t_j-t_{j+1})+\log\Theta_j \tag{9.5}$$

Rearranging the sum on the right of (9.5) 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 $\min_m E_{Q_{\mathbf{b}}}l_m\le\phi^*$ then $\min_m E_{Q_{\mathbf{b}}}l_m\le top(\mathbf{c})$ with equality only if $top(\mathbf{c})=\phi^*$. Now $\mu_k=t_k-\min_m E_k l_m$ so $\mu_k\ge 0$ only if $t_k=\phi^*$. But this is just the stopping condition. If there is no stopping then all $\mu_j>0$ and

$$\log(g_{k+1})\le\log(N)+\sum_{j=1}^{k}[\Delta_j(t_j-t_{k+1})-\mu_j^2/2B^2] \tag{9.6}$$

Since $\log(g_{k+1})\ge 0$ the sum on the right of (9.6) must be bounded below.

Take a subsequence {k'} such that $t_{k'+1} \to \limsup t_k = \bar{t}$ and look at (9.6) along this subsequence assuming $\bar{t} = \phi^* + \delta$ where $\delta > 0$. Let $N_{k'}$ be the number of terms in the sum in (9.6) that are positive. We claim that $\sup_{k'} N_{k'} < \infty$. To show this, suppose the jth term is positive, i.e.

$$t_j > t_{k'+1} + \mu_j^2 / 2B^2 \tag{9.7}$$

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 (9.7) 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 (9.6) 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

$$\log(g_{k'+1}) \leq S + \sum_{j'} (\varepsilon \bar{\Delta} - \mu_{j'}^2 / 2B^2) \tag{9.8}$$

Take $\varepsilon \leq \delta / 2$. For k' large enough and all j', $t_{j'} > \phi^* + \delta/2$ and $\mu_{j'}^2 / 2B^2 \geq \delta^2 / 8B^2$. Taking ε so that $\varepsilon \bar{\Delta} < \delta^2 / 16B^2$ shows that the number of terms in the (9.6) 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 $\limsup t_k = \phi^*$.

10. About the Optimizing Values

10. 1 Finding the minimizing Δ.

With a zero-one loss, as in classification, the minimizing Δ is given by a simple expression. By its definition

.

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

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

If the loss function is not zero-one finding the minimizing Δ involves a search for the minimum of a one-dimensional convex function. The suboptimal value $\Delta_k=\mu_k/B^2$ can be used and the convergence proof will still hold. However, I suspect convergence may be considerably slower.

10.2 The optimal **c**, Q

For optimal **c**, Q

$$\min_m E_Q l_m = top(\mathbf{c}) = \phi^*$$

In general there are many optimal **c** and Q and there is no simple characterization of them. However, in the two class classification situation, we can give a heuristic argument that indicates that the optimal Q tend to concentrate their weight near the boundary between classes.

If p(j|**x**), j=1,2 are the probabilities of classes 1,2 at the point x then the ε−boundary between the class is defined by:

$$BD_\varepsilon = \{\mathbf{x}; |p(1|\mathbf{x}) - p(2|\mathbf{x})| \leq \varepsilon\}$$

If N is large, in the instances $(y_n,\mathbf{x}_n)\varepsilon T$ such that $\mathbf{x}_n \varepsilon BD_\varepsilon$ there are roughly equal numbers such that y_n =1 and y_n =2. Also, these two groups of instances are dispersed almost at random over BD_ε. For Q concentrated on the instances such that $\mathbf{x}_n \varepsilon BD_\varepsilon$, if the set $\{h_m\}$ of classifiers do not contain classifiers that can separate instances randomly dispersed on BD_ε. then

$$\min_m E_Q l_m \cong (1/2) - \varepsilon$$

for some small ε >0 and this is about as large as $\min_m E_Q l_m$ can get.

Intuitively, it is clear why Q would tend to concentrate on the points near the boundary. These are the hardest points to classify correctly and Player I will do well to focus his strategy on these points.

10.3 Does $\mathbf{c}^{(k)}$ converge?

If the non-decreasing $|\mathbf{b}^{(k)}|$ sequence has a finite limit, then it is not difficult to show that the sequence of vectors $\mathbf{c}^{(k)}$ converges. But if the $|\mathbf{b}^{(k)}|$ go to infinity, then the situation is unclear. The set of optimal **c** form a convex, closed set of vectors in M-space and it is possible that the $\mathbf{c}^{(k)}$ sequence circulates endlessly through this convex set.

11. An Optimizing Arcing Algorithm for the Convex Regression Game

The matrix game gives upper bounds for the convex classification and regression games. A natural question is whether there exist optimizing arcing algorithms for the original convex games. There is such an algorithm for regression. Recall that

$$rs(\mathbf{z}_n, \mathbf{c}) = (y_n - \sum c_m h_m(\mathbf{x}_n))^2$$

$$= (\sum c_m (y_n - h_m(\mathbf{x}_n))^2$$

$$= (\sum c_m r_{mn})^2 \qquad (11.1)$$

where $r_{mn}=y_n-h_m(\mathbf{x}_n)$., and that

$$\phi^*=\min_{\mathbf{c}} top(\mathbf{c})=\max_Q \min_{\mathbf{c}} E_Q rs(\mathbf{z},\mathbf{c}) .$$

Here is the description of the optimizing arcing algorithm arc-gv . Initialize by setting $Q_1(\mathbf{z}_n)=1/N$. Let m_1 be the minimizer of $\sum_n r_{mn}^2$. Set $b_{m_1}^{(1)}=1$, with the other components of $\mathbf{b}^{(1)}$ zero. Take B such that $r_{mn}^2<B$, all m,n and $\overline{\Delta}$ any number much larger than 1/B. For any M-vector **v,** denote

$$(\mathbf{v},\mathbf{r})_n=\sum_m v_m r_{mn}$$

Definition 11.1 *arc-gv updates* $\mathbf{b}^{(k)}$ *as follows:*

1) Let

$$Q_k(\mathbf{z}_n)\sim\exp((\mathbf{b}^{(k)},\mathbf{r})_n^2-t_k|\mathbf{b}^{(k)}|^2)$$

ii) Let m_{k+1} *be the minimizer of over m of*

$$E_{Q_k}((\mathbf{c}^{(k)},\mathbf{r})_n r_{mn}) \qquad (11.2)$$

If the minimum value of (11.2) is $\le t_k$ *then stop.*

iii) Let Δ_k *be the minimizer over the interval* $[0,\bar{\Delta}]$ *of*

$$E_{Q_k} \exp(2\Delta[(\mathbf{b}^{(k)},\mathbf{r})_n r_{mn} - t_k|\mathbf{b}^{(k)}|] + \Delta^2(r_{mn}^2 - t_k))$$

where $m = m_{k+1}$.

iv) Increase $b_{m_{k+1}}^{(k)}$ *by* Δ_k *and continue.*

The difference between this arcing algorithm and the arcing algorithms for the matrix game is in (11.2). Instead of minimizing the Q-expectation of the fixed loss function $l(y_n,h_m(\mathbf{x}_n))$, we are minimizing an error function thast depends on the current value of **c**.

Theorem 11.2 *If arc-gv stops at the kth step, then* $top(\mathbf{c}^{(k)})=\phi^*$. *Otherwise* $\lim_k top(\mathbf{c}^{(k)})=\phi^*$.

proof. Let $\mathbf{c}=\mathbf{b}/|\mathbf{b}|$, $t=top(\mathbf{c})$ and define

$$w(\mathbf{z}_n,\mathbf{b}) = \exp((\mathbf{b},\mathbf{r})_n^2 - t|\mathbf{b}|^2)$$

$$g(\mathbf{b}) = \sum_n w(\mathbf{z}_n,\mathbf{b})$$

$$Q_{\mathbf{b}}(\mathbf{z}_n) = w(\mathbf{z}_n,\mathbf{b})/g(\mathbf{b})$$

Increase the mth coordinate of **b** by the amount x/|**b**| getting **b**'. Let

$$\lambda(m,\mathbf{b},x,\mathbf{z}_n) = \exp(2x((\mathbf{c},\mathbf{r})_n r_{mn} - t) + x^2(r_{mn}^2 - t)/|\mathbf{b}|^2)$$

The following identity is not difficult to verify:

$$g(\mathbf{b}') = \exp(|\mathbf{b}'|^2(t-t'))\sum_n \lambda(m,\mathbf{b},x,\mathbf{z}_n) w(\mathbf{z}_n,\mathbf{b}) \qquad (11.3)$$

Taking logs

$$\log g(\mathbf{b}') = \log g(\mathbf{b}) + |\mathbf{b}'|^2(t-t') + \log\Theta(m,\mathbf{b},x) \qquad (11.4)$$

where

$$\Theta(m,\mathbf{b},x) = E_{Q_{\mathbf{b}}} \lambda(m,\mathbf{b},x,\mathbf{z})$$

Proposition 11.3 If

$$\min_m E_{Q_{\mathbf{b}}}((\mathbf{c},\mathbf{r})_n r_{mn} \geq top(\mathbf{c}) \tag{11.5}$$

then $top(\mathbf{c})=\phi^*$ *and* $\min_m E_{Q_{\mathbf{b}}}((\mathbf{c},\mathbf{r})_n r_{mn}=\phi^*$.

proof: Assume (11.5) holds. Then

$$top(\mathbf{c}) \leq \min_m E_{Q_{\mathbf{b}}}((\mathbf{c},\mathbf{r})r_m = \inf_{\mathbf{c}'} E_{Q_{\mathbf{b}}}((\mathbf{c},\mathbf{r})(\mathbf{c}',\mathbf{r})) \leq$$

$$\sqrt{E_{Q_{\mathbf{b}}}(\mathbf{c},\mathbf{r})^2}\sqrt{E_{Q_{\mathbf{b}}}(\mathbf{c}',\mathbf{r})^2} \leq \sqrt{top(\mathbf{c})}\sqrt{\phi^*}$$

Proposition 11.4 *If*

$$top(\mathbf{c}) - E_{Q_{\mathbf{b}}}((\mathbf{c},\mathbf{r})r_m = \mu_{\mathbf{b}} > 0$$

there is a constant D>0 depending only on B and $\bar{\Delta}$ *such that for x in the interval* $[0, \bar{\Delta}/|\mathbf{b}|]$

$$\min_x \Theta(m,\mathbf{b},x) \leq 1-\mu_{\mathbf{b}}^2/D \tag{11.6}$$

proof: To simplify notation, let $\Theta(x)=\Theta(m,\mathbf{b},x)$. A partial expansion gives

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

Let $\sigma_n=2(\mathbf{c},\mathbf{r})_n r_{mn}-2t$ and $\gamma_n=2(r_{mn}^2-t)/|\mathbf{b}|^2$. Then

$$\Theta''(\alpha x) \leq \max_n(\gamma_n+(\sigma_n+x\gamma_n)^2)\Theta(\alpha x)$$

For $x\leq \bar{\Delta}/|\mathbf{b}|$

$$\max_n(\gamma_n+(\sigma_n+x\gamma_n)^2) \leq \max_n \gamma_n + 2\max_n \sigma_n^2 + 2\bar{\Delta}^2 \max_n \gamma_n^2|\mathbf{b}|^2$$
$$\leq 2B+4B^2(1+2\bar{\Delta}^2)=2D$$

So in the specified x-range,

$$\Theta(x)\leq 1-2\mu_{\mathbf{b}}x+Dx^2\Theta(\alpha x).$$

Let [0,s] be the largest interval such that on this interval $\Theta(x)\leq 1$ In this range

$$\Theta(x)\leq 1-2\mu_{\mathbf{b}}x+Dx^2 \qquad (11.7)$$

The right hand side of (11,7) has a minimum at $x^*=\mu_{\mathbf{b}}/D$ Clearly, x*<s and $x^*/|\mathbf{b}|\leq 2B/D\leq 1/B\leq\overline{\Delta}$.

At the kth step of arc$-\phi^*$ let $x_k=\Delta_k/|\mathbf{b}^{(k)}|$. From identity (11.4) and using the same notation as in the proof of convergence for the matrix game,

$$\log g_{k+1}=\log g_k+|\mathbf{b}^{(k+1)}|^2(t_k-t_{k+1})+\log(\Theta_k) \qquad (11.8)$$

Summing (11.8) gives

$$\log g_{k+1}=\log g_1+\sum_{j=1}^{k}|\mathbf{b}^{(j+1)}|^2(t_j-t_{j+1})+\log\Theta_j.$$

Rearrange the sum and using (11.6) gives

$$\log g_{k+1}\leq\log g_1+\sum_{j=1}^{k}((|\mathbf{b}^{(j+1)}|^2-|\mathbf{b}^{(j)}|^2)(t_j-t_{j+1})-\mu_j^2/D) \qquad (11.9)$$

From (11.9) the rest of the proof goes exactly as in the matrix game proof.

12. Are Small Margins and VC-bounds the Explanation?

In their 1997 paper Schapire et.al. explain the success of Adaboost in lowering generalization error to its success in lowering margins and showed that it produced generally lower margins than bagging. In this section we give empirical results that show that this explanation is not valid and that the explanation must lie deeper. Furthermore, the results cast doubt on ability of VC-type bounds to give an adequate qualitative picture of what factors influence generalization error.

The Schapire et.al. generalization error bound, on which their explanation is founded, is based on the distribution of the margin on the training set and the VC-dimension of the class of classifiers employed. For fixed VC-dimension, the smaller the margins, the smaller the generalization error bound. The bound stated in Section 4 has a similar implication--for a given

VC-dimension, the lower top($\mathbf{c}$), the lower the bound on the generalization error. If the Schapire et.al. explanation is correct, and the VC-type bounds are giving the right qualitative picture, then for VC-dimension fixed, the size of the margins and of top($\mathbf{c}$) are the determining factors in the size of the generalization error.

To do an empirical check on this, 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 missclassification cost. Setting the trees selected to have k terminal nodes fixes the VC-dimension.

Then, 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 small. Adaboost is not touted to do a maximal reduction of the margins, hence should not, by theory to date, produce as low as generalization error as arc-gv. To check this, 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 $mgn(\mathbf{z}_n,\mathbf{c})=2*er(\mathbf{z}_n,\mathbf{c})-1$.

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 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($\mathbf{c}$) and these values were also averaged over the 10 run 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 except for the heart data (also described in Breiman[1997]). The real data sets have the following number of input variables and instances--heart 6-1395: breast cancer 9-699: ionosphere 34-351: diabetes 8-768. 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 larger data sets so that tree sizes would be appropriate to the data set.

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	6.0	5.7	24.0	31.0
k=16	5.8	5.2	9.8	25.5
threenorm				
k=8	20.2	18.8	35.5	37.2
k=16	18.3	18.2	20.0	28.4
ringnorm				
k=8	6.7	6.1	26.6	32.1
k=16	7.1	5.3	11.5	23.0
heart				
k=32	1.4	1.4	23.8	26.9
k=64	1.1	0.4	8.4	15.8
breast cancer				
k=16	3.6	3.1	16.7	21.1
k=32	5.1	3.4	1.9	8.1
ionosphere				
k=8	3.7	3.7	21.3	21.4
k=16	7.4	3.1	4.1	14.1
diabetes				
k=32	25.7	27.0	32.3	33.3
k=64	25.1	25.8	17.5	22.6

Although the test set errors for arc-gv and adaboost are generally close, the pattern is that adaboost has a test set error slightly less than that of arc-gv. The diabetes data set is a bit different, but is known to contain errors and outliers. 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.

A last issue is whether lower values of top(**c**) translate into generally lower values of the margin. That is, it might happen that although top(**c**) was lower, the average, say, of mgn($\mathbf{z}_n$,**c**) increased. We looked at this by graphing er(**c**,$\mathbf{z}_n$) for the first run of twonorm (k=16) using arc-gv vs. the values of er(**c**,$\mathbf{z}_n$) for the similar run using adaboost. The same data was used in both runs.

The results are shown in Figure 1. In only 6 instances out of 300 are the er($\mathbf{z}_n$,**c**) values resulting from adaboost larger than those from arc-gv. In the remaining instances, they are significantly smaller. Figure 2 is a similar graph for the first run of ionosphere with k=16. Here, every value of er($\mathbf{z}_n$,**c**) for vg is less than its value for adaboost.

The conclusion is that these drastically different margin distributions for the same VC-dimension had little effect on the generalization error. In fact, that smaller margins usually led to high er($\mathbf{z}_n$,$\mathbf{c}$) generalization error. But this is the opposite of what is indicated by the VC-type bounds.

12. Clues to the Mystery

12.,1 Concentration on Instances

Since theoretical foundations for the efficiency of various arcing algorithms are shaky, we are in a clue hunting mode. One interesting clue is provided by looking at how concentrated the distributions Q($\mathbf{z}_n$) become. At the end of each run of 100 iterations, the information $I(Q)=-\sum_n Q(\mathbf{z}_n)\log Q(\mathbf{z}_n)$ is computed. This quantity is averaged over the 10 runs then exponentiated, div9ided by N and multiplied by 100 to give IE(Q). The interpretation of IE(Q) is as the percentage of training instances that Q is concentrated on. For instance, if Q is uniform on all instances, then IE(Q)=100. If Q is uniformly concentrated on K instances then IE(Q)=100K/N. Table 2 gives the values of IE(Q) for the various data sets and values of k, repeating the values of top($\mathbf{c}$) for reference.

Table 2 Values of IE(Q) and top($\mathbf{c}$)x100

	IE(Q)		top($\mathbf{c}$)	
data set	arc-gv	adaboost	arc-gv	adaboost
twonorm				
k=8	43	23	24.0	31.0
k=16	62	21	9.8	25.5
threenorm				
k=8	53	43	35.5	37.2
k=16	67	38	20.0	28.4
ringnorm				
k=8	46	32	26.6	32.1
k=16	60	26	11.5	23.0
heart				
k=32	47	31	23.8	26.9
k=64	50	19	8.4	15.8
breast cancer				
k=16	12	6	16.7	21.1
k=32	7	3	1.9	8.1
ionosphere				
k=8	26	15	21.3	21.4
k=16	28	8	4.1	14.1
diabetes				
k=32	47	36	32.3	33.3
k=64	54	28	17.5	22.6

Thus, the Q for adaboost is concentrated on far fewer instances then is arc-gv. Further, in most data sets, as the complexity is increased by increasing k, arc-gv concentrates on more instances while adaboost concentrates on fewer. The original intuitive idea behind adaboost was that it concentrated weight on instances most likely to be missclassified. Work in Breiman[1997] showed that there were other algorithms, sharing the property that they concentrated weight on hard-to-classify instances, and resulting in generalization errors comparable to those of adaboost.

But the evidence in Table 2 indicates that arc-gv, in its effort to produce lower values of top(**c**), concentrates on too many instances--more than just the hard-to-classify. Still, given the wide disparity in the IE(Q) values, it is curious that the estimated generalization errors of the two algorithms are so close.

12.2 Circulating Around the Inner Rim

Conceptualize the graph of top(**c**) versus the M-dimensional vector **c** as a large valley such that at the bottom lies the convex set of **c** such that top(**c**) has its minimum value. The arc-gv algorithm takes appropriately small steps going in size to zero; reaches the bottom and walks slowly around in it.

Other arcing algorithms, such as adaboost and arc-x4, take much larger steps, circulate around the inner rim of the valley, and do not reach the bottom. These algorithms are surprisingly resistant to overfitting, For instance, the estimated generalization error of adaboost either stays the same or gets smaller as k increases. On the other hand, the error of arc-gv is sometimes significantly increased when a larger k is used.

This indicates that skating around the inner rim instead of slogging to the bottom usually gives lower errors. But it certainly raises the question of why this is so and how far down the rim one ought to be for optimal performance.

13 Randomizing the Paths

An alternative randomized version of each arcing algorithm is gotten by generating the probability Q_k as in the original algorithm. Then form a new training set T' of size N by sampling, *with replacement*, N times from the instances in T, where the nth instance $\mathbf{z}_n$ has probability $Q_k(\mathbf{z}_n)$ of being drawn at each of the N draws. In the original algorithm, there is an error measure err(h) such that the h_m selected minimizes the Q_k weighted value of err(h). In the randomized version, h_m is selected to minimize the equi-weighted value of err(h). The increase in the **b**-weight is computed the same way.

The randomized version of arc-gv was run on the synthetic data sets--the three used above and the 21-dimensional 300 instance, 3-class wave form data. It was run on the real data sets used above plus the 6-class glass data: 9-214, and the 19 class soybean data:35-683 (both in the UCI repository). There were ten runs on each data set, with 200 iterations on each synthetic data set and 100 on the real. More iterations were used on the synthetic data because the test error continued to decrease a bit from 100 to 200 iterations.

These results were compared to running a randomized version of Adaboost. The values of top(**c**) and IE(Q) at the end of each run were also recorded. The summary is given in Table 3.

Table 3 Test Set Error(%), IE(Q), top(**c**)x100

Data Set	Test Error	IE(Q)	top(**c**)
waveform			
adaboost	16.5	31	24.2
arc-gv	17.2	68	17.0
twonorm			
adaboost	4.0	20	20.7
arc-gv	4.4	64	13.1
threenorm			
adaboost	17.1	34	24.5
arc-gv	17.5	76	17.8
ringnorm			
adaboost	5.2	30	22.5
arc-gv	5.7	68	15.0
heart			
adaboost	0.9	18	18.3
arc-gv	0.6	44	9.6
breast cancer			
adaboost	3.4	5	15.9
arc-gv	3.7	16	6.7
ionosphere			
adaboost	6.6	13	19.6
arc-gv	5.4	34	10.7
diabetes			
` adaboost	22.1	31	25.6
arc-gv	22.6	56	20.2
glass			
adaboost	18.6	34	26.9
arc-gv	18.6	56	21.1
soybean			
adaboost	7.5	6	12.1
arc-gv	8.2	7	9.8

The curious aspect of the randomized algorithms is that no regularization is needed. The trees combined were grown as large as possible and no pruning was used. In past experiments, I found that this worked better than using any pruning. In contrast, some limits have to be placed on the size of the trees in the deterministic algorithm to prevent overfitting. The table above shows that the indications of overfitting by the deterministic verion of arc-gv does not appear in the randomized version.

The test set performance of the two algorithms is very similar. Adaboost does sightly better on the synthetic data. On the six real data sets they have exactly the same total test set error. The surprising aspect is that the test set results are so close, while the differences in top(**c**) and IE(Q) are so marked. I conjecture that randomising the path gets arc-gv away from the bottom of the top(**c**) valley so it also skates around the inner rim, but at as lower level than Adaboost. But the small effect of the top(**c**) and IE(Q) differences is mystifying.

14. Remarks

There is a 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 less efficient than the arc-gv method. I have not been able to find an arcing algorithm that performs optimally for the convex classification game. The $\Theta(\Delta)$ function is only piece-wise continuous and it seems difficult to bound it below one.

In the sequel to this paper, implementations of the arc-gv algorithm in regression will be studied. Drucker[1997] has applied adaboost ideas to regression in an ad hoc fashion and gotten generally better results than bagging. There are two versions of arc-gv for regression. One for the dominating matrix game and the other for the original convex game. Both will be implemented and tested.

15. 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.

References

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

Breiman, L. [1997] 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

Drucker, H. [1997] Improving Regressors using Boosting Techniques, Proceedings of the Fourteenth International Conference on Machine Learning, pp. 107-115..

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

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

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 I Proof of Theorem 4.1

Our proof is patterned after the proof given in Schapire et.al.[1997] of a similar result based on the margin distribution. Denote $l(m,\mathbf{z})=l_m(\mathbf{z})$. Let K to be a positive integer and fixing **c** take J_k^* k=1, ... ,K to be independent random variables such that $P(J_k^*=m)=c_m$. Denote by **J*** the random K-vector whose kth component is J_k^* Conditional on **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 $\mu<\lambda$

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

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

$$E(P(\bar{L}(\mathbf{Z})<\mu,er(\mathbf{Z},\mathbf{c})\geq\lambda|\mathbf{Z}))\leq E(P(\bar{L}(\mathbf{Z})-E(\bar{L}(\mathbf{Z})|\mathbf{Z})<\mu-\lambda|\mathbf{Z})) \qquad \text{(AI.2)}$$

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

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

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

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

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

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

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

$$\exp(-N(P\{P(\bar{L}(\mathbf{Z},\mathbf{j})\geq\mu)-I(\bar{L}(\mathbf{Z},\mathbf{j})\geq\mu)\}\geq\varepsilon)) \quad \text{(AI.6)}$$

We can upper bound the term in the exponent of (AI.6).

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

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

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

Hence, (AI.4) is bounded by $M^k\exp(-\varepsilon N)$. Take a grid of M values μ_i , i=1,... ,M equispaced in [0,B]. 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, write

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

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

The last term in (AI.7), assuming $\nu<\mu$, is bounded by

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

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

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

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

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

Let the ε in (AI.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 $\leq \delta_K$, for all $\lambda, v, \mathbf{c}$, and for fixed K,

$$P(er(\mathbf{Z},\mathbf{c}) \geq \lambda) \leq \varepsilon_K + S_K + I(top(\mathbf{c}) > v) \qquad \text{(AI.9)}$$

Take $\delta_K = 2^{-K}\delta$. Then (AI.9) also holds for all K except on a fixed set of training sets with probability $\leq \delta$. Now let $v = top(\mathbf{c}), \lambda = \Delta + top(\mathbf{c}), \sigma = 8(B/\Delta)^2$, and take $K = \sigma\log(2N^2/\sigma\log(2M))$. Take Δ so that $(\Delta/B) \geq \sqrt{8/M}$, then $2N \geq \exp(K/2M)$ and if $R = \sigma\log(2M)/N$ then

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

Fgure 1

Comparison of er(z)-twonorm data, k=16

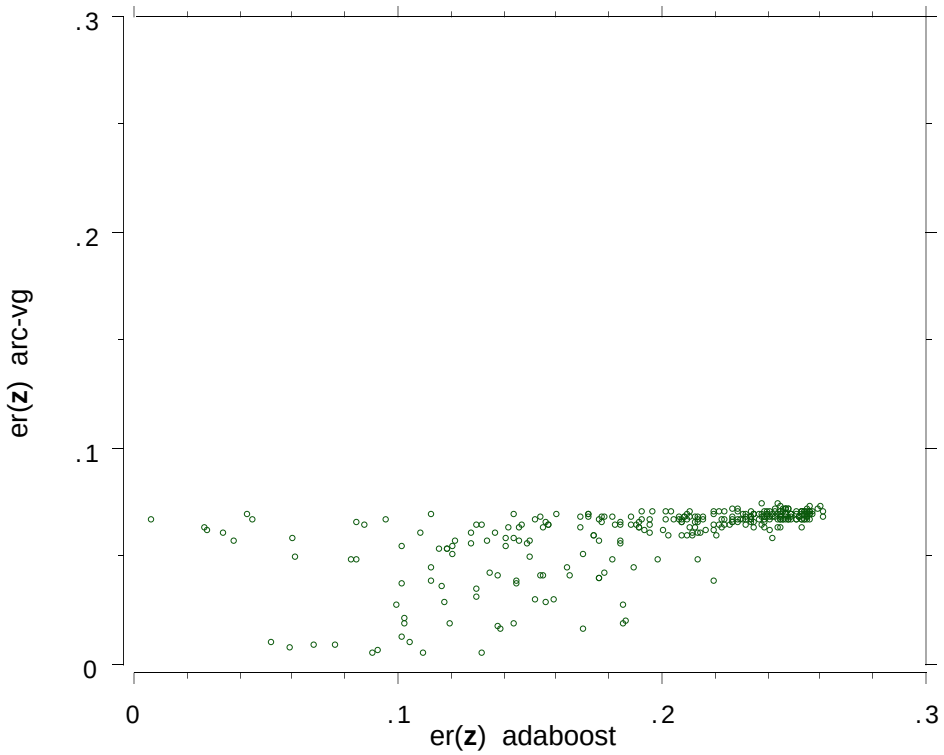

Figure 2

Comparison or er(z)-ionosphere data, k=16