# Hinging Hyperplanes for Regression, Classification, and Function Approximation

Leo Breiman

***Abstract*—A hinge function $y=h(x)$ consists of two hyperplanes continuously joined together at a hinge. In regression (prediction), classification (pattern recognition), and noiseless function approximation, use of sums of hinge functions gives a powerful and efficient alternative to neural networks with compute times several orders of magnitude less than fitting neural networks with a comparable number of parameters. The core of the methodology is a simple and effective method for finding good hinges.**

***Index Terms*—Nonlinear function approximation, classification, prediction, regression, hyperplanes.**

## I. Introduction

In an $M$-dimensional space $(x_1, \cdots, x_M)$, a *hinge function* $y = h(\boldsymbol{x})$ consists of two hyperplanes continuously joined together. Taking $x_0 \equiv 1$ and using $\cdot$ to denote the inner product of two vectors, if the two hyperplanes are given by

$$y = \beta^+ \cdot \boldsymbol{x}, \qquad y = \beta^- \cdot \boldsymbol{x},$$

they are joined together on $\{\boldsymbol{x}\colon (\beta^+ - \beta^-) \cdot \boldsymbol{x} = 0\}$ and we refer to $\Delta = \beta^+ - \beta^-$, or any multiple of $\Delta$, as the *hinge* for the function. The explicit form of the hinge function is either $\max(\beta^+ \cdot \boldsymbol{x}, \beta^- \cdot \boldsymbol{x})$ or $\min(\beta^+ \cdot \boldsymbol{x}, \beta^- \cdot \boldsymbol{x})$.

Most of the recently introduced methods for nonlinear regression, classification, and function approximation use expansions into sums of basis functions. The basis functions used are "data selected" from a large parametric class of primitive functions. For instance, CART (Breiman *et al.* [5]) uses an expansion into indicator functions of multidimensional rectangles with sides parallel to the coordinate axes. Neural network methods use sigmoid functions of linear functions as primitives. The MARS method (Friedman [8]) uses products of univariate linear spline functions as its primitive class. In this work, the hinge functions form the primitive class. There are good reasons, as given below, for this approach.

Let $P$ be any measure with compact support on $E^{(M)}$ and $f(\boldsymbol{x})$ any sufficiently smooth function. Then we show in Section III, using methods developed by Jones [11] and Barron [1] that there is a constant $C(f, P)$ such that for any $K$, there are hinge functions $h_1, \cdots, h_K$ with

$$\left\| f - \sum_1^K h_k \right\|^2 \leq \frac{C}{K}.$$

The property that makes the hinge functions effective is that there is a simple and computationally efficient method for locating hinges. Suppose we are told that $y = h(\boldsymbol{x})$ is a hinge function with unknown hinge $\Delta^*$; are given data $(y_n, \boldsymbol{x}_n)$, $n = 1, \cdots, N$; and want to use this data to locate the hinge. One approach is this: for any specified candidate hinge $\Delta$, do a least squares fit of a hinge function with hinge $\Delta$ to the data. Let RSS($\Delta$) be the residual sum-of-squares. Now search over $\Delta$-space to find the minimizer of RSS($\Delta$). This procedure is computationally intensive and global searches are not feasible unless $M$ and $N$ are small.

Here is an alternative: start with an arbitrary hinge $\Delta^{(0)}$. Using least squares, fit the data on the side $\Delta^{(0)} \cdot \boldsymbol{x}_n \geq 0$ to a hyperplane $y = \beta^+ \cdot \boldsymbol{x}$, and do a similar fit to the data such that $\Delta^{(0)} \cdot \boldsymbol{x}_n < 0$ getting $y = \beta^- \cdot \boldsymbol{x}$. Take the new estimate for the hinge as

$$\Delta^{(1)} = \beta^+ - \beta^-,$$

and repeat the procedure, getting a sequence $\Delta^{(k)}$ of estimates.

In Section II, we give evidence that generally $\Delta^{(k)} \to \Delta^*$ and that the convergence is rapid. The noisy case, $y_n = h(\boldsymbol{x}_n) + \epsilon_n$, is also looked at and the accuracy of $\lim \Delta^{(k)}$ as an estimate of $\Delta^*$ examined. Simulations support the theoretical accuracy results and show that the hinge finding algorithm is computationally efficient and accurate even for large $M$, and high noise.

In Section III, we state the theorem regarding the approximation of a smooth function by a sum of hinge functions. Then, we look at the implementation that at the $K$th stage adds a new hinge function by using the hinge finding algorithm on $y = f(\boldsymbol{x}) - \sum_1^{K-1} h_k(\boldsymbol{x})$, and then readjusts the sum $\sum_1^K h_k(\boldsymbol{x})$. Simulations in dimensions ranging from $M = 2$ to $M = 16$ show good approximation properties and verify the $1/K$ decrease.

The most important applications of hinge functions are to multivariate regression and classification. Section IV discusses the use of hinge functions to produce a nonlinear prediction function given noisy data $\{(y_n, \boldsymbol{x}_n),\ n = 1, \cdots, N\}$. Examples are given to show how sums of hinge functions can be used to construct accurate predictors. In particular, to cope with high dimensional spaces, a variable selection method in hinge finding is introduced. It took 2.8 minutes of cpu

Manuscript received November 18, 1991; revised July 27, 1992. This work was presented in part at Neural Networks in Computing, Snowbird, UT, March 1992.

The author is with the Department of Statistics, University of California at Berkeley, Berkeley, CA 94720.

IEEE Log Number 9208640.

time to compute the prediction equation in a highly nonlinear 100-dimensional example with a training set of size 2000.

In Section V, we show how the hinge finding algorithm can be extended so that hinge functions can be usefully employed in classification (pattern recognition) problems. The approach is to formulate the $J$-class problem in terms of $J$-linked regression equations, and then to locate, at each stage, the hinge that is optimum for a combination of the equations. This leads to a $J \times J$ eigenvalue problem at each iteration. Examples are given, with both real and simulated data to illustrate the effectiveness of this method.

Our concluding remarks and summary appear in Section VI, and the proofs of theorems are in Appendix A. Computational aspects of the hinge procedures are discussed in Appendix B. Appendix C gives some equations for computing cross-validated error measure in regression and classification.

On current computing equipment, the procedures are quite feasible for substantial sample sizes and dimensionality and we give cpu times involved for various examples in the text. The examples were run on a variety of machines. To provide uniform time benchmarks, the 16-dimensional example of Section III was run on all machines including an IBM RS6000/540. The timings given in the text are scaled to the RS 6000, a 12.5 megaflop machine.

Another important application of hinge finding is in determining splits for the construction of classification and regression trees. This is the subject of a sequel paper.

For recent advances in constructing nonlinear prediction functions, see (in the statistical literature) Freidman and Stuetzle [9], Breiman *et al.* [5], Brieman and Friedman [6], Friedman [8], Hastie and Tibshirani [10], Breiman [2], Wahba [15]. In classification, some recent statistical publications are Breiman *et al.* [5], Breiman and Ihaka [7], Hastie and Tibshirani [10]. In the engineering and computer science fields, where recent attention has been focussed on neural networks, see Lippmann [13] for a review and references.

## II. Finding the Hinge

### A. Analytic Results

Assume that $y = h(\boldsymbol{x})$ is a hinge function with unknown hinge, and that there is a distribution of points $\boldsymbol{x} \in E^{(M+1)}$ governed by a probability measure $P(d\boldsymbol{x})$. For a fixed vector $\Delta^{(0)}$, denote

$$S_+ = \{\boldsymbol{x}\colon \Delta^{(0)} \cdot \boldsymbol{x} \geq 0\}, \quad S_- = \{\boldsymbol{x}\colon \Delta^{(0)} \cdot \boldsymbol{x} < 0\}$$

$$\Gamma_+(m, m') = \int_{S_+} x_m x_{m'}\, dP, \quad \Gamma_-(m, m') = \int_{S_-} x_m x_{m'}\, dP$$

$$(xy)_+(m) = \int_{S_+} x_m h(\boldsymbol{x})\, dP, \quad (xy)_-(m) = \int_{S_-} x_m h(\boldsymbol{x})\, dP.$$

The least-squares coefficients of a hyperplane fitted to the $y$-values in $S_+$ are $\beta_+ = \Gamma_+^{-1}(xy)_+$, and those in the $S_-$ fit are $\beta_- = \Gamma_-^{-1}(\boldsymbol{x}y)_-$. The new hinge value $\Delta^{(1)}$ is $\beta_+ - \beta_-$ and the process is repeated starting from $\Delta^{(1)}$.

Since $y = h(\boldsymbol{x})$ is a hinge function, there is a hinge $\Delta^*$ such that for $S_+^* = \{\boldsymbol{x}\colon \Delta^* \cdot \boldsymbol{x} \geq 0\}$ and $S_-^* = \{\boldsymbol{x}\colon \Delta^* \cdot \boldsymbol{x} < 0\}$, if the fitted coefficients on the $+$, $-$ sides are $\beta_+^*$, $\beta_-^*$, then $\Delta^* = \beta_+^* - \beta_-^*$. Thus,

$$(\boldsymbol{x}y)_+(m) = \int_{S_+ \cap S_+^*} x_m(\beta_+^* \cdot \boldsymbol{x})\, dP + \int_{S_+ \cap S_-^*} x_m(\beta_-^* \cdot \boldsymbol{x})\, dP$$

$$(\boldsymbol{x}y)_-(m) = \int_{S_- \cap S_+^*} x_m(\beta_+^* \cdot \boldsymbol{x})\, dP + \int_{S_- \cap S_-^*} x_m(\beta_-^* \cdot \boldsymbol{x})\, dP,$$

Denote

$$\Gamma_{++}(m, m') = \int_{S_+ \cap S_+^*} x_m x_{m'}\, dP$$

with the analogous definitions for $\Gamma_{+-}$, $\Gamma_{-+}$, $\Gamma_{--}$. Then, some manipulations give

$$\Delta^{(1)} = A\Delta^*,$$

where

$$A = I - \Gamma_+^{-1}\Gamma_{+-} - \Gamma_-^{-1}\Gamma_{-+}. \tag{2.1}$$

*Hinge Convergence Theorem 1:* Denote by $s$ any linear function of $x_0, \cdots, x_M$. Assume that $P$ satisfies

a) There is a $c < \infty$ such that for every $s$ with $Es^2 \leq 1$, $Es^4 \leq c$.
b) Let $p(\alpha) = \sup_{Es^2=1} P(|s| \geq \alpha)$. Then, $p(\alpha) = O(\alpha)$ as $\alpha \to 0$.
c) If $S_+ = \{\boldsymbol{x};\ s \geq 0\}$ and $P(S_+) = c > 0$, the inf over $s$ of the minimum eigenvalue of $\Gamma_+$ is $\lambda^*(c) > 0$.

Then, there is a $\delta > 0$ such that if $\|\Delta^{(0)} - \Delta^*\| \leq \delta$, then

$$\|\Delta^{(k)} - \Delta^*\| \to 0.$$

and convergence is exponentially fast.

The proof is in Appendix A. Both theoretical and simulation results suggest that convergence occurs even for $\Delta^{(0)}$ distant from $\Delta^*$.

*Theorem 2:* If $x_0 \equiv 0$ and $(x_1, \cdots, x_M)$ have a joint normal distribution, then $\Delta^{(k)} \to \Delta^*$ for any starting $\Delta^{(0)}$ such that $\Delta^{(0)} \cdot \Delta^* \neq 0$. The proof is given in Appendix A.

### B. Simulations

The simulations were done as follows: the $(\Delta^*(1), \cdots, \Delta^*(M))$ were taken to be $M$ independent and identically distributed (i.i.d.) $N(0, 1)$'s, and $\Delta^*(0)$ adjusted so that the proportion of data in $\{\Delta^* \cdot x \geq 0\}$ was uniform on [0.15, 0.85]. Initial hinge values $\Delta^{(0)}$ were selected by taking $(\Delta^{(0)}(1), \cdots, \Delta^{(0)}(M))$ to be another $M$-vector of i.i.d. $N(0, 1)$'s and $\Delta^{(0)}(0)$ taken so that $\#\{\boldsymbol{x}_n\colon \Delta^{(0)}\boldsymbol{x}_n \geq 0\} = \text{int}(N/2)$ (here $\{\cdot\}$ denotes the cardinality of the set $\{\cdot\}$). Starting from $\Delta^{(0)}$, the hinge finding algorithm was run to

convergence giving $\Delta$. The run was counted as a hit if $|\cos(\boldsymbol{\Delta}, \Delta^*)| \geq 0.99$, otherwise a miss. The whole procedure, including the data generation, was repeated 100 times.

In the first run, $N = 250$, the dimension $M = 6$, and $(x_1, \cdots, x_M)$ were sampled from $X_1, \cdots, X_M$ with $X_m = Z_m - Z_{m-1}$, $\{Z_0, \cdots, Z_M\}$ independent unit exponentials. To study the affect of dimensionality, the same structure was used in the second run but with $M = 12$. Since this example used long tailed distributions, in the next two runs we again set $X_m = Z_m - Z_{m-1}$ but took the $\{Z_m\}$ to be uniform on [0, 1]. The first run was at dimension 6 and the second at dimension 12.

There were no misses in the 400 repetitions. But the algorithm can be made to miss. For instance, at least 13 points are needed to define a hyperplane in 12 dimensions. We set $\Delta^*(0)$ in the 12-dimensional exponential case so that $\#\{\boldsymbol{x}_n;\ \Delta^* \mathbf{x}_n \geq 0\} = 20$. Now 20 points to define a corner hinge in 12 dimensions is sparse. In 100 trials, there were 13 misses.

### C. The Effect of Noise

Suppose that the data is of the form $\{y_n, \boldsymbol{x}_n\}, n = 1, \cdots, N$ with $y_n = h(\boldsymbol{x}_n) + \epsilon_n$, $h(\boldsymbol{x}_n)$ a hinge function and $\{\epsilon_n\}$ noise. Then,

$$\Delta^{(k+1)} = \Delta^* - B\Delta^* + \Gamma_+^{-1}(\epsilon x)_+ - \Gamma_-^{-1}(\epsilon x)_-$$

where $B$ is $\Gamma_+^{-1}\Gamma_{+-} + \Gamma_-^{-1}\Gamma_{-+}$. The matrix $B$ is a function $B(\Delta^{(k)}, \Delta^*)$. The proof of the hinge convergence theorem indicates that $B$ is small order of the noise. That is, if the noise is small, and $\Delta^{(k)} \sim \Delta^* + Z$, where $Z$ is a linear function of the noise, then $B \sim o(Z)$. Thus, for small noise, at convergence

$$\Delta \cong \Delta^* + \Gamma_+^{-1}(\epsilon x)_+ - \Gamma_-^{-1}(\epsilon x)_-.$$

From this,

$$E\|\Delta - \Delta^*\|^2 = \frac{\sigma^2}{N}(\mathrm{tr}(\Gamma_+^{-1}) + \mathrm{tr}(\Gamma_-^{-1})). \tag{2.3}$$

### D. Simulations: Noisy Case

The $\{\boldsymbol{x}_n\}$, $n = 1, \cdots, N$ data was generated as in the first example in Section II-B. Then $y_n$ was set equal to $h(\boldsymbol{x}_n) + \epsilon_n$ with the $\{\epsilon_n\}$ i.i.d. $N(0, \sigma^2)$ noise. The hinge $\Delta^*$ was selected as in Section II-A.

The signal/noise ratio (s/n) is here (also see Table I) defined by first fitting a least-squares hyperplane to $\{h(\boldsymbol{x}_n)\}$. Then, the signal $\sigma_l$ is defined as the square root of the mean residual sum-of-squares (MRSS). The noise is defined by its standard deviation $\sigma_N$. Then $s/n = \sigma_l/\sigma_N$. Another way to look at this is that the expected MRSS on fitting $h + \epsilon$ by the right hyperplane is $\sigma_l^2 + \sigma_N^2$. The expected MRSS on fitting by the right hinge function is $\sigma_N^2$. The ratio of expected MRSS is

$$\frac{\text{MRSS(hinge)}}{\text{MRSS(linear)}} = \frac{1}{1 + (\text{s/n})^2}.$$

The s/n ratios used were 1, 1/2, 1/3, 1/4. The output of each run consisted of the averages over the repetitions of

TABLE I
EFFECTS OF NOISE

| s/n | av(\|cos\|) | Sd(\|cos\|) | av($\|\Delta - \Delta^*\|/D$) | Sd($\|\Delta - \Delta^*\|/D$) | RSSRATIO |
|---|---|---|---|---|---|
| 1 | 0.99 | 0.01 | 0.92 | 0.38 | 1.00 |
| 1/2 | 0.93 | 0.07 | 1.11 | 0.58 | 1.00 |
| 1/3 | 0.81 | 0.20 | 1.31 | 0.67 | 1.00 |
| 1/4 | 0.68 | 0.24 | 1.49 | 0.71 | 0.99 |

$|\cos(\Delta, \Delta^*)|$ and of $\|\Delta - \Delta^*\|/D$ where

$$D = \sqrt{\frac{\hat{\sigma}^2}{N}(\mathrm{tr}(\Gamma_+^{-1}) + \mathrm{tr}(\Gamma_-^{-1}))},$$

with $\Gamma_+$, $\Gamma_-$ computed using $\Delta$ and

$$\hat{\sigma}^2 = \frac{1}{N - 2(M+1)}\sum_n (y_n - \hat{f}(\boldsymbol{x}_n))^2,$$

where $\hat{f}$ is the estimated hinge function. The discussion leading to (2.3) indicates that the expectation of $\|\Delta - \Delta^*\|/D$ should be close to one in low noise situations. The standard deviation of these two quantities over the repetitions in the run is also given.

Another quantity of interest is $\hat{\sigma}^2/(\sum_i^N \epsilon_n^2/N)$. This is the ratio of the MRSS using the estimated hinge function debiased by the factor $1/(N-2(M+1))$, to the MRSS using the known ridge function. Denote this ratio by RSSRATIO. We report on four runs of 100 repetitions each using $N = 250$, and the six-dimensional exponential distribution specified previously.

Its surprising how well the algorithm does even with high noise levels. When s/n = 0.25, only a 6% decrease in RSS is gotten by fitting the underlying hinge instead of a hyperplane. Yet the values of the RSSRATIO show that the minimum RSS hinge is consistently being found.

## III. NOISELESS FUNCTION APPROXIMATION

On $E^{(M)}$ a sum of hinge functions $\sum_1^K h_k(\boldsymbol{x})$ is a contionuous piecewise linear function. Results of Barron [1] on sigmoid function and Jones [11] on sinusoids can be extended to show that sums of hinge functions are effective in function approximation. Let $\tilde{f}(w)$ be the Fourier transform of $f(\boldsymbol{x})$.

*Theorem 3:* If the support of $P$ is contained in the sphere of radius $R$, and if

$$\int \|w\|^2 |\tilde{f}(\boldsymbol{w})|\, d\boldsymbol{w} = c < \infty,$$

then there are hinge functions $h_1, \cdots, h_K$ such that

$$\left\| f - \sum_1^K h_k \right\|^2 \leq \frac{(2R)^4 c^2}{K}.$$

The proof of this theorem is an extension of Barron's extension of the Jones result and is given in Appendix A. In the Jones and Barron results, a function $\phi$ on $E^{(1)}$ is specified, and the approximating sums are of the form $\sum_1^K \phi(\beta_k \cdot \boldsymbol{x})$. In neural network approximation, the sum which minimizes $\|f(\boldsymbol{x}) - \sum_1^K \phi(\beta_k \cdot \boldsymbol{x})\|$ is gotten by using local gradient

searches to minimize over the $\{\beta_1, \cdots, \beta_K\}$. Not only is this highly computationally intensive even for $K$, $M$ of moderate size, but only local minima are guaranteed.

Barron and Jones show that there is a "greedy" algorithm which at each step enters the next function $\phi(\beta_K \cdot \boldsymbol{x})$ by minimization over $M+2$ parameters only, and still achieves the upper bound of the existence proofs.

A similar "greedy" result can be proved for hinge functions but the point may be moot. The hinge finding algorithm makes the optimization over the entire sum computationally efficient. Our simulations show that the decrease in squared norm is inversely linear but that the constant is orders of magnitude less than that given in Theorem 3.

### A. The Approximation Algorithm

The basic algorithm is: given a function specified at the points $\{\boldsymbol{x}_n\}$, $n = 1, \cdots, N$ run the hinge finding algorithm on this data, resulting in a hinge function approximation $h(\boldsymbol{x})$. Since the function to be fitted is not a hinge function, $M$ starting values of $\Delta^{(0)}$ are used, and the hinge adopted is that with minimum RSS. The $m$th starting $\Delta^{(0)}$ is given by $\Delta^{(0)}(j) = 0$, $j \neq m$, $\Delta^{(0)}(m) = 1$, and $\Delta^{(0)}(0)$ selected so that the condition $\{\Delta^{(0)} \cdot \boldsymbol{x} \geq 0\}$ cuts the data in half.

To compute the approximation to $f(\boldsymbol{x})$ using $K$ hinge functions, at the $K$th stage find the hinge function approximation $h_K(\boldsymbol{x})$ to $f(\boldsymbol{x}) - \sum_1^{K-1} h_k(\boldsymbol{x})$. Then refit: update $h_1$ by refitting the difference $f - \sum_2^K h_k$. Using this updated $h_1$, update $h_2$ by refitting $f - h_1 - \sum_3^K h_k$. After $h_K$ is refitted, start the cycle again with $f_1$. These cycles are continued until there is no further appreciable decrease in RSS. The procedure is made more efficient by using, as the *single* starting $\Delta^{(0)}$ for each refit, the current hinge of the function to be refitted.

Another algorithm was also tested. Each hinge function is the sum of a linear function and a function $h^+(\Delta \cdot \boldsymbol{x})$ where $h^+(x) = x$, $x \geq 0$, and $0$, $x < 0$. Let $h^+(\Delta_k \cdot \boldsymbol{x})$, $k = 1, \cdots, K-1$ be the nonlinear parts of the hinge functions entered at steps $1, 2, \cdots, K-1$. Do a linear regression of $f(\boldsymbol{x})$ on the $M+K$ variables $1, x_1, \cdots, x_M$, $h^+(\Delta_1 \cdot \boldsymbol{x}), \cdots, h^+(\Delta_{K-1} \cdot \boldsymbol{x})$, getting

$$\hat{f}_{K-1}(\boldsymbol{x}) = \sum_0^M \beta_m x_m + \sum_{k=1}^{K-1} \gamma_k h^+(\Delta_k \cdot \boldsymbol{x}).$$

Use the hinge finding algorithm on $f - \hat{f}_{K-1}$ to find $h_K$ and suppose the nonlinear part of $h_K$ is $h^+(\Delta_K \cdot \boldsymbol{x})$. Do a linear regression of $f(\boldsymbol{x})$ on the $M+K+1$ variables

$$1, x_1, \cdots, x_M, h^+(\Delta_1 \cdot \boldsymbol{x}), \cdots, h^+(\Delta_K \cdot \boldsymbol{x})$$

getting new coefficients $\beta_0, \cdots, \beta_M$ and $\gamma_1, \cdots, \gamma_K$. Take $\hat{f}_K$ to be the linear combination using the new coefficients.

Although refitting is computationally fairly efficient, the cpu time to fit, say, 50 hinge functions in 16 dimensions to 1000 data points is considerably larger than the regression type algorithm described in the above paragraph. The trade off in cpu time and accuracy is explored in the simulations reported on in the next section.

### B. Simulations: Function Approximation

The first example is chosen for visual inspection. There are 1000 $\{\boldsymbol{x}_n\}$ values uniformly distributed on the square $[0, 1]^2$. Let $e = (1, 1)$, then

$$f(\boldsymbol{x}) = e^{-\|\boldsymbol{x}-0.3e\|^2} - e^{-\|\boldsymbol{x}-0.7e\|^2}.$$

Fig. 1(a) gives the surface plot of $f$. Hinge functions are fitted and refitted. The surface plots of $\sum_1^K h_k(x)$, $K = 4, 8, 16, 32$ are given in Fig. 1(b)–(e).

We are not advocating the use of hinge functions to fit smooth surfaces in low dimensions. Other methods are available which give smoother and more accurate fits in two or three dimensions (see, for instance, Breiman [1], Friedman [8], Wahba [15]). This two-dimensional example is given only because visual inspection is possible.

The second series of examples are similar except for dimension. The function is

$$f(\boldsymbol{x}) = e^{-\|\boldsymbol{x}\|^2/2}.$$

The points $\{\boldsymbol{x}_n\}$ are distributed on the sphere $\|\boldsymbol{x}\| \leq 3$ using a spherically symmetric distribution such that $\|\boldsymbol{x}\|$ is uniform on $[0, 3]$. For this function,

$$M = \int \|w\|^2 |\tilde{f}(w)| \, d\boldsymbol{w},$$

so that the upper bound of theorem 3.1 is $1296M^2/K$.

For $M = 4$, the refit and regression algorithm were run up to $K = 50$. Fig. 2 gives the plots of $1/\|f - \sum_1^K h_k\|^2$ vs. $K$ for both the refit and regression algorithms for $K = 1$ to 50. Fitting the $1/\|f - \sum_1^K h_K\|^2$ data by $K/b$ using least squares gives $b = 0.022$(refit) and $b = 0.061$(regression). Either one is orders of magnitude smaller than the constant given by Theorem 3.

There is a third set of points plotted in Fig. 2. The suspicion arises that if one put down a series $\Delta_1, \cdots, \Delta_K$ of hinges chosen at random and regressed $f(\boldsymbol{x})$ on the variables $1, x_1, \cdots, x_M$, $h^+(\Delta_1 \cdot \boldsymbol{x}), \cdots, h^+(\Delta_K \cdot \boldsymbol{x})$ one might do almost as well as using the hinge finding algorithm. To check this, random hinges $\Delta_1, \cdots, \Delta_K$ were generated with $(\Delta_K(1), \cdots, \Delta_K(M))$ being i.i.d. $N(0, 1)$ and $\Delta_K(0)$ taken such that the proportion of data satisfying $\{\Delta_K \cdot \boldsymbol{x}_n \geq 0\}$ is uniform on [0.1, 0.9]. The values $\|f - \sum_1^K h_k\|^{-2}$ generated this way form the third graph on Fig. 2.

To explore the effect of dimensionality, we ran the refit algorithm using dimensions 4, 8, 16 and going up to $K = 50$. The data was generated as previously described with sample size 1000. Fig. 3 gives the graphs of $1/\|f - \sum_1^K h_k\|^2$ vs. $K$ for $K = 1, \cdots, 50$, including the results for dimension four. The MRSS for dimension 16 decreases rapidly when more than 20 hinge functions are fitted. This is probably due to the fact that 1000 data points are sparse in 16 dimensions. Fitting 20 hinge functions involves optimizing over almost 400 parameters, and 50 hinge functions, over almost 900. As the number of parameters approach the number of data points, the error of the *fit at these points* drops rapidly to zero.

A more interesting comparison of the effect of dimensionality is when the number of parameters is constrained to be

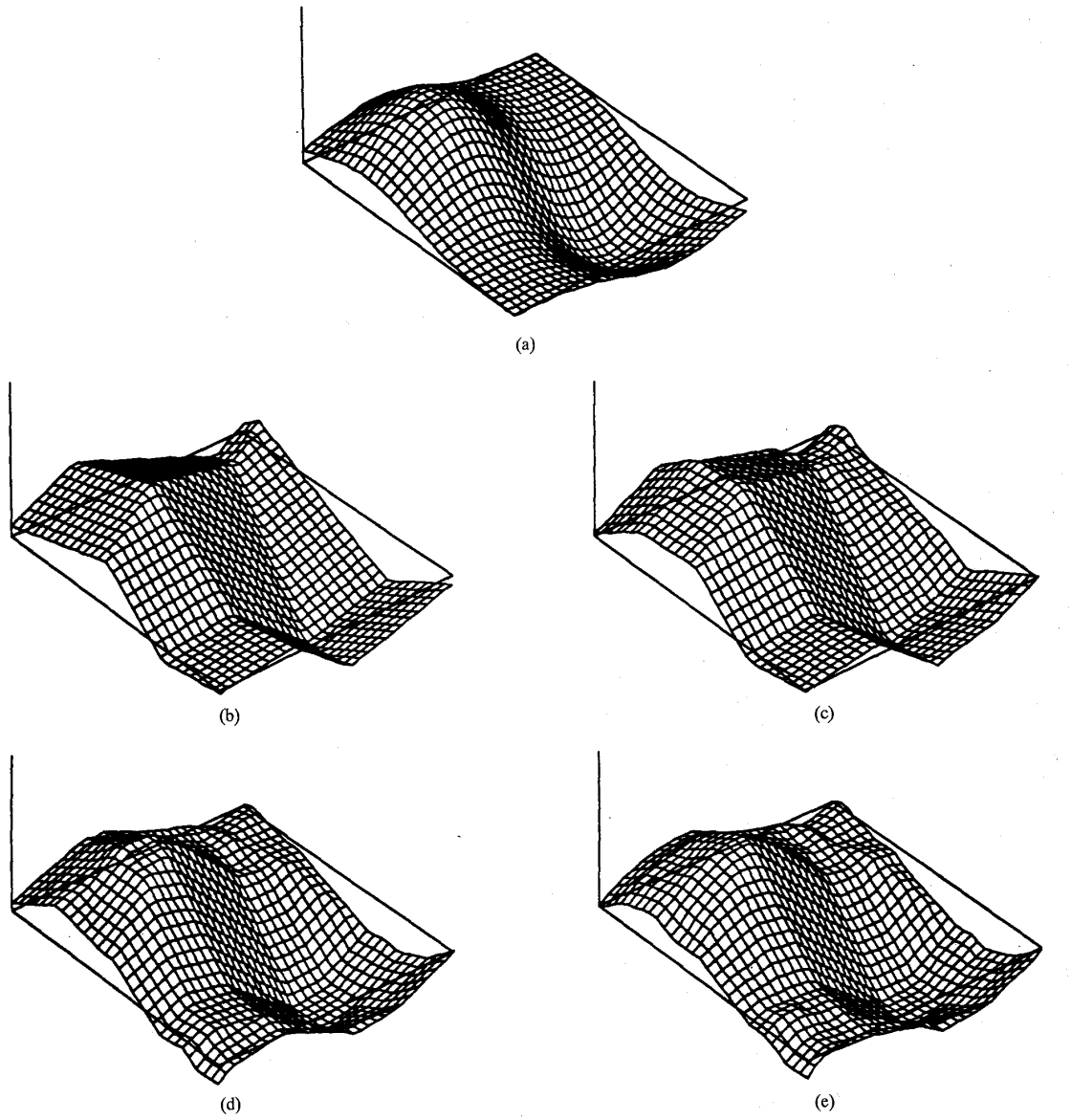


Fig. 1. Surface plots of hinge approximations. (a) Original function. (b) 4 hinges. (c) 8 hinges. (d) 16 hinges. (e) 32 hinges.

small compared with the number of data points. Fig. 4 graphs the same data as Fig. 3 but only up to 20 hinge functions. There is very little effect due to dimensionality.

The run doing fitting and refitting of the 50 hinge functions in 16 dimensions took 11.4 cpu minutes (RS 6000), pretty fast for a highly nonlinear optimization involving almost 900 parameters. The regression type algorithm took 2.3 cpu minutes.

## IV. Using Hinge Functions in Regression

Hinge functions give a potentially powerful new approach to nonlinear regression. Given data of the form $\{(y_n, \boldsymbol{x}_n),\ n = 1, \cdots, N\}$ the problem is to use this data to construct a function $\hat{f}(\boldsymbol{x})$ that will give accurate predictions of future unknown $y$-values when the corresponding $x$-values are known.

If $\hat{f}(\boldsymbol{x})$ is restricted to be linear, then squared-error loss leads to classical least squares regression. If the number of variables is nominal, say $\leq 20$, then the methods proposed by Friedman [8] and Breiman [2] give continuous predictors $\hat{f}(\boldsymbol{x})$ restricted to be the sum of nonlinear functions each depending on only a few variables, say one, two, or at most three. But with some data, predicting $\boldsymbol{y}$ by sums of functions of a small number of the $x$-variables may not give accurate results. Consider, for instance, the function $e^{-\|\boldsymbol{x}\|^2}$ in $M$-dimensions.

Methods for fitting continuous functions to high interaction, high-dimensional data are rare. One early and remarkable result due to Meisel and Collins [14] derives a piecewise

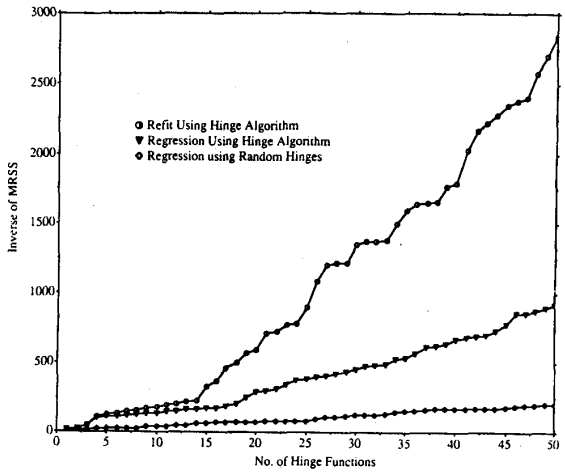


Fig. 2. Comparison of methods in four dimensions.

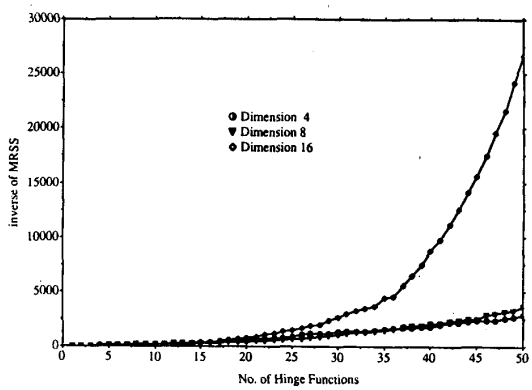


Fig. 3. Inverse MRSS using refitting.

continuous hyperplane estimate using a method much different from hinging. Friedman and Stuetzle [9] originated "projection pursuit" regression which uses a sum of estimated smooth functions of linear functions. The tree-structured approach (Breiman *et al.* [5]) fits a discontinuous histogram-like functions.

The approach using hinge functions in prediction is the same as in the noiseless case—find the best fit to $y$ of $\sum_1^K h_k(\boldsymbol{x})$. One important issue is how many hinge functions to fit. If $K$ is too large, overfitting occurs and the fit loses accuracy on future data. Two methods are useful for selecting $K$. The simplest uses the $\mathrm{PE}_{\mathrm{GCV}}$ criterion:

Let MRSS $(K)$ be the mean residual sum-of-squares. The $\mathrm{PE}_{\mathrm{GCV}}$ estimate for the test set prediction error is given by

$$\mathrm{PE}_{\mathrm{GCV}}(K) = \mathrm{MRSS}(K)/(1 - c(K+1)(M+1)/N)^2,$$

where $N$ is the sample size, and $M$ the dimension and $c$ a parameter in the [1, 3] range. Now take $K$ to minimize $\mathrm{PE}_{\mathrm{GCV}}(K)$. The most accurate estimate of test set error is given by cross-validation, which requires much more computing. For derivations see Appendix C.

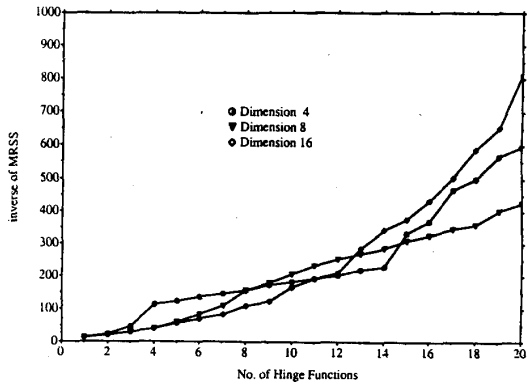


Fig. 4. Inverse MRSS using refitting.

### A. A High Interaction Example

As an example of a high interaction problem in 10-dimensional space, take

$$g(\boldsymbol{x}) = \mathrm{logit}(v_1(\boldsymbol{x})) + \mathrm{logit}(v_2(\boldsymbol{x})) + \mathrm{logit}(v_3(\boldsymbol{x})),$$

where $\mathrm{logit}(\boldsymbol{x}) = e^x/(1+e^x)$. The $v_1(\boldsymbol{x})$, $v_2(\boldsymbol{x})$, $v_3(\boldsymbol{x})$ are linear in $(x_1, \cdots, x_{10})$ defined by letting

$$l_1 = 10x_1 + 9x_2 + 3x_3 + 7x_4 - 6x_5 - 5x_6 - 9x_7 - 3x_8 - 2x_9 - x_{10}$$

$$l_2 = -x_1 - 2x_2 - 3x_3 - 4x_4 - 5x_5 - 6x_6 + 7x_7 + 8x_8 + 9x_9 + 10x_{10}$$

$$l_3 = -x_1 - 2x_2 - 3x_3 + 4x_4 + 5x_5 + 4x_6 - 3x_7 - 2x_8 - x_9.$$

The $\{\boldsymbol{x}_n\}$ are selected from a uniform distribution on $[0, 1]^{10}$. Normalize the $l_1$, $l_2$, $l_3$ to have upper 97.5% point = 2.0, and then put $v_i$ equal to $2(l_i - 2)$.

Sample size was 400 and the $\{y_n\}$ generated as

$$y_n = a \cdot g(\boldsymbol{x}_n) + \epsilon_n,$$

where the $\{\epsilon_n\}$ are unit normal noise. The constant $a$ was defined so that for $f(\boldsymbol{x}_n) = a \cdot g(\boldsymbol{x}_n)$, the standard deviation of the $\{f(\boldsymbol{x}_n)\}$ was 4.0, giving a s/n ratio of 4.0.

A 4000 member test set was generated and used to estimate the prediction error ($\mathrm{PE}_{\mathrm{TS}}$) as successive hinge functions were added. The $\mathrm{PE}_{\mathrm{GCV}}$ for $c = 1.5$ was also computed. The initial value of the sample variance of the $\{y_n\}$ was 17.4. A linear regression fitted to the data resulted in a mean residual-sum-of-squares of 10.3.

The number of hinge functions fitted was increased from 1 up to 7 with the results summarized in Table II. The coefficients of the hinges in the 3-hinge fit are given in Table III. The action of these three hinges is clear from inspection. The first hinge is fitting $\mathrm{logit}(v_2)$ on one side and the other two logits on the other side. The second hinge is fitting $\mathrm{logit}(v_1)$ on one side and the other two logits on the other side. Similarly,

TABLE II
SUMMARY OF FIT BY HINGE FUNCTIONS

| # Hinge Functions | MRSS | $PE_{GCV}$ | $PE_{TS}$ |
|---|---|---|---|
| 1 | 5.22 | 6.20 | 5.69 |
| 2 | 2.07 | 2.69 | 2.49 |
| 3 | 1.09 | 1.56 | 1.52 |
| 4 | 1.00 | 1.58 | 1.52 |
| 5 | 0.96 | 1.59 | 1.53 |
| 6 | 0.82 | 1.62 | 1.54 |
| 7 | 0.78 | 1.75 | 1.59 |

TABLE III
COEFFICIENTS OF HINGES IN 3-HINGE FIT

| Variable | First Hinge | Second Hinge | Third Hinge |
|---|---|---|---|
| 1 | −2.2 | 9.8 | 0.4 |
| 2 | −6.0 | 10.6 | 0.5 |
| 3 | −8.1 | 8.2 | 1.4 |
| 4 | 10.9 | 6.8 | 3.5 |
| 5 | 13.4 | −5.7 | 5.9 |
| 6 | 10.9 | −5.2 | 6.3 |
| 7 | −9.0 | −5.0 | −6.6 |
| 8 | −6.5 | −3.1 | −7.4 |
| 9 | −3.1 | −2.6 | −6.8 |
| 10 | 0.7 | −2.5 | −8.2 |

the third hinge fits logit($v_3$) on one side and the others on the other side.

The minimum $PE_{GCV}$ selection criterion picks the same 3-hinge fit as the minimum $PE_{TS}$ criterion, (although $PE_{TS}$ ties between the 3- and 4-hinge fits). In fact, the $PE_{GCV}$ is a minimum at the 3-hinge fit for every value of the parameter $c$ in [1], [3].

The running time for this example is 7.4 cpu seconds.

### *B. Stepwise Forward Selection of Variables and a Higher Dimensional Examples*

If the data set has, say, 1000 cases and 100 variables, then fitting 4 hinge functions involves the estimation of 500 parameters. This is only 2 cases per parameter estimated and will probably result in a noisy estimate. If possible, one would want to keep a tighter control on the number of parameters estimated. In addition, with larger dimensionality, fitting hinge functions becomes slower.

To deal with these two issues, a stepwise forward variable selection method for entering hinge functions is used. Here is the idea: start with a search through all $M$ variables to find the single variable hinge that gives minimum RSS. Call the variable used $x_1$. Now search among all $x_m$, $m \geq 2$ to find the lowest RSS hinge based on the pairs of variables $(x_1, x_m)$. Keep adding variables until a minimum is found in the generalized cross-validation (GCV) estimate of prediction error. Then start this process over to find the next hinge: refit and keep adding hinges until the $PE_{GCV}$ hits a minimum.

To illustrate, the following 100-dimensional example was run: the sample size was 2000 and the $\{\boldsymbol{x}_n\}$ were sampled from the uniform distribution on $\{0, 1\}^{100}$. Define

$$g(\boldsymbol{x}) = \sum_1^K \text{logit}(v_i(\boldsymbol{x})),$$

where the $\{(v_i(\boldsymbol{x})\}$ are linear functions of $x_1, \cdots, x_{100}$. The $y_n$ are given by

$$y_n = a \cdot g(\boldsymbol{x}_n) + \epsilon_n,$$

where the $\{\epsilon_n\}$ are unit normals and $a$ is taken to make the s/n rate 4.0.

The $\{v_1\}$ are defined as follows: For $k = 20, 40, 60, 80$, let

$$c(k, m) = \exp(-0.7|m - k|),$$

define

$$u_k(\boldsymbol{x}) = \sum_m c(k, m)x_m,$$

and

$$\begin{aligned} l_1 &= 3u_1 - u_2 - u_3 - u_4 \\ l_2 &= -u_1 + 3u_2 - u_3 - u_4 \\ l_3 &= -u_1 - u_2 + 3u_3 - u_4 \\ l_4 &= -u_1 - u_2 - u_3 + 3u_4. \end{aligned}$$

Normalize the $l_i$ to have unit variance and take $v_i = l_i - 1$.

The sample variance of the $\{y_n\}$ was 17.1. A linear regression on all 100 variables gives a mean residual-sum-of-squares of 16.3, so the regression surface was predominantly nonlinear. The time needed to run this example was 2.8 cpu minutes (RS 6000). Table IV summarizes the results.

TABLE IV
SUMMARY OF RESULTS FOR THE 100-DIMENSIONAL EXAMPLE

| Hinge | # of variables* | MRSS | $PE_{GCV}$** | $PE_{TS}$ |
|---|---|---|---|---|
| 1 | 21 | 11.01 | 11.78 | 12.70 |
| 2 | 20 | 6.71 | 7.67 | 8.20 |
| 3 | 25 | 2.83 | 3.52 | 3.57 |
| 4 | 13 | 2.16 | 2.82 | 2.69 |
| 5 | 4 | 2.13 | 2.83 | 2.71 |
| 6 | 5 | 2.09 | 2.84 | 2.73 |
| 7 | 3 | 2.08 | 2.86 | 2.75 |
| 8 | 3 | 2.06 | 2.88 | 2.79 |
| 9 | 3 | 2.04 | 2.90 | 2.83 |
| 10 | 4 | 2.02 | 2.90 | 2.85 |

*The minimum number of variables in a hinge was set at 3.
**$c = 1.5$.

The structure of both of these examples was devised so that an optimum fit could be gotten using a neural network. It would be interesting to see what accuracies and compute times are produced by running these examples on a neural network program.

## V. Using Hinges in Classification

To use hinges in classification, the problem needs to be reformulated into a regression context. Suppose there are $J$ classes numbered $1, \cdots, J$ with the probability $P(j|\boldsymbol{x})$ of being in class $\boldsymbol{x}$ assumed known. Then the Bayes optimal classification rule is: classify $\boldsymbol{x}$ into that class $j$ for which $P(j|\boldsymbol{x})$ is maximum.

Let the random variable $Y$ be one in class $j$, otherwise zero. Then the function $\theta$ of the vector $\boldsymbol{X}$ that minimizes $E(Y - \theta(\boldsymbol{X}))^2$ is $P(j|\boldsymbol{x})$. With classification type data $\{(j_n, \boldsymbol{x}_n),\ n = 1, \cdots, N\}$ and $j_n \in \{1, \cdots, J\}$, these remarks suggest the following approach: define

$$y_{jn} = \begin{cases} 1, & j_n = j, \\ 0, & \text{otherwise.} \end{cases}$$

Find $J$ functions $\hat{\theta}_j(\boldsymbol{x})$ such that $\text{RSS}_j = \|\boldsymbol{y}_j - \hat{\theta}_j\|^2$ is small. Classify a future object with observed $\boldsymbol{x}$ as class $j$ if

$$\theta_j(\boldsymbol{x}) = \max_i \theta_i(\boldsymbol{x}).$$

The problem is now to estimate the functions $\theta_j(\boldsymbol{x})$ from the data. If linear $\theta_j$ are used, the result is similar to discriminant analysis, and performance can be poor in situations where the $P(j|\boldsymbol{x})$ are not well approximated by linear functions. The neural network methodology uses $\theta_j(\boldsymbol{x})$ estimated through the use of sums of sigmoid functions of linear functions.

One approach is—for each $j$ approximate the $\{y_{jn}\}$ by a sum of hinge functions $\sum_1^K h_{jk}(\boldsymbol{x}_n)$. This uses up parameters at an alarming rate. A different approach, employed in classification trees, for instance, is to use a common basis of functions, $h_1, \cdots, h_K$, to approximate each set of $\{y_{jn}\}$, with the approximations differing from class to class through the coefficients of the $\{h_k\}$, i.e., the approximations

$$\boldsymbol{y}_j \sim \sum_k \alpha_{jk} h_k(\boldsymbol{x}) \tag{5.1}$$

are used. Then the problem becomes to find those hinge functions that make

$$\text{RSS} = \sum_j \min_{\{\alpha_{jk}\}} \|\boldsymbol{y}_j - \sum_{K=1}^{K} \alpha_{jk} h_k\|^2$$

as small as possible.

Suppose $h_1, \cdots, h_{K-1}$ have been selected and residuals

$$r_{jn} = y_{jn} - \sum_{k=1}^{K-1} \alpha_{jk} h_k(x_n)$$

computed. What is wanted is a hinge function $f_K$, and coefficients $\{\gamma_j\}$ to minimize

$$\sum_{j,n} (r_{jn} - \gamma_j f_K(\boldsymbol{x}_n))^2. \tag{5.2}$$

Start with a hinge $\Delta^{(0)}$, and denote by $(+)$, that data for which $\Delta^{(0)}\boldsymbol{x} \geq 0$ and $(-)$ for the other data. Consider fitting a hyperplane $\beta^+ \cdot \boldsymbol{x}$ to the $(+)$ data, and a hyperplane $\beta^-$ to the $(-)$ data and selecting $\{\gamma_j\}$, $\beta^+$, $\beta^-$ to minimize

$$\sum_{-,j} (r_{jn} - \gamma_j \beta^- \cdot x_n)^2 + \sum_{+,j} (r_{jn} - \gamma_j \beta^+ \cdot x_n)^2. \tag{5.3}$$

After this minimization has been carried out, the next hinge is $\Delta^{(1)} = \beta^+ - \beta^-$, and the process is repeated until convergence.

Minimization of (5.3) leads to either a $J \times J$ or $M \times M$ eigenvalue problem. We derive the $J \times J$ problem. To begin note that there is an indeterminancy in the scales of $\gamma$, $\beta^+$, $\beta^-$. This is resolved by taking $\|\gamma\|^2 = 1$. Partial derivatives of (5.2) with respect to $\beta_m^+$, $\beta_m^-$ leads to the equations

$$S^+\beta^+ = Z^+\gamma,$$
$$S^-\beta^- = Z^-\gamma,$$

where $S^+$ is the $M \times M$ matrix, $S^+_{mm'} = \sum_+ x_{mn}x_{m'n}$, $Z^+$ is the $M \times J$ matrix $Z^+_{mj} = \sum_+ x_{mn}r_{jn}$; similarly for $S^-$ and $Z^-$. Substituting these expressions for $\beta^+$, $\beta^-$ into (5.3) gives the expression

$$\sum_{j,n} r_{jn}^2 - \sum_{j,j'} \gamma_j \gamma_{j'} H(j, j'),$$

where

$$H = (Z^+)^t (S^+)^{-1} Z^+ + (Z^-)^t (S^-)^{-1} Z^-.$$

To minimize the RSS we want to maximize $\gamma^t H \gamma$ under the constraint $\|\gamma\| = 1$. This solution is the eigenvector of $H$ corresponding to its largest eigenvalue. Once $\gamma$ is known, then $Z^+\gamma$, $Z^-\gamma$ are computed, $\beta^+ = (S^+)^{-1}Z^+\gamma$, $\beta^- = (S^-)^{-1}Z^{-1}\gamma$ and $\Delta^{(1)} = \beta^+ - \beta^-$. After $h_K$ is entered, then refitting cycles are carried out refitting both the $\{h_k\}$ and the $\{\alpha_{jk}\}$. The decision on how many hinge functions to use in the fit can be based on a test set or cross-validation.

### A. Two Examples

The first data set used as an example was provided to us by Richard Lippmann. To quote him "The database consists of the first two formants frequencies (F1 and F2) of vowels (Peterson and Barney, "Control methods used in a study of vowels," *The Journal of the Acoustical Society of America*, vol. 24, no. 2, pp. 175–184, Mar. 1952) in Hz. The actual data was gathered by digitizing a figure from Rabiner and Schafer, *Digital Processing of Speech* (Prentice-Hall, 1978). The digitized data was arbitrarily divided into 338 training samples and 333 testing samples."

There are ten vowel classes having roughly equal representations in both the training and test set. This two dimensional data was used as a benchmark for various classifiers by Lee and Lippman [11]. Table V in an excerpt of their results.

Fitting 22 hinge functions gave a test set error of 18.6% and training set error of 20.4%. The compute time was 41.9 cpu seconds. This translates into 517.5 cpu seconds scaled to the machine used by Lee and Lippman. Although we conceived of the hinge function methodology as primarily useful in high dimensions, it is competitive in this two-dimensional example.

### B. Simulated Wave Form Data

The second example consists of simulated data with structure given in pp. 49–55 of the book by Breiman *et al.* [5]. It

TABLE V
CLASSIFIER PERFORMANCE

| Classifier | Test Set | % Error Training Set | Times (cpu seconds) |
|---|---|---|---|
| Unimodal Gaussian | 20.4 | 22.5 | 31.4 |
| Back-Prop. | 21.0 | 21.0 | 2458.4 |
| Hypersphere (RCE) | 23.1 | 17.3 | 144.9 |
| KNN | 17.4 | 20.4 | .5 |
| LVQ | 18.0 | 13.5 | 425.9 |
| Feature Map | 19.5 | 22.2 | 113.1 |
| CART Tree | 21.9 | 16.0 | 8.2 |

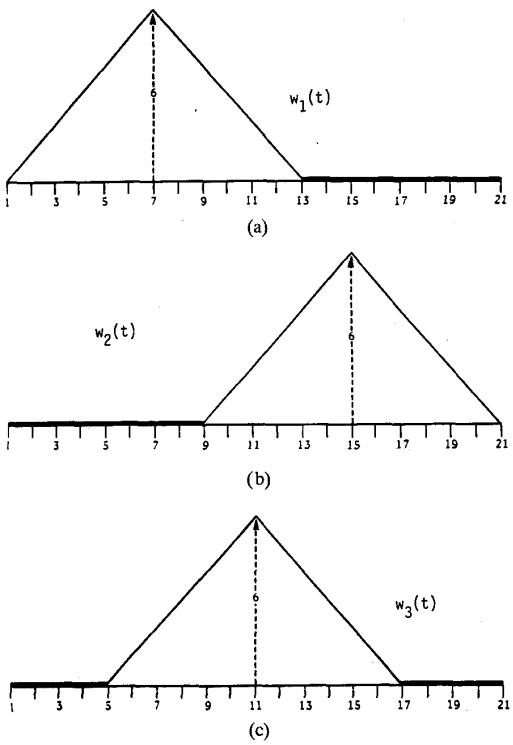


Fig. 5. Waveforms. (a) $w_1(t)$. (b) $w_2(t)$. (c) $w_3(t)$.

is a 3-class, 21-dimensional problem based on the waveforms $w_1(t)$, $w_2(t)$, $w_3(t)$ graphied in Fig. 5.

Each class consists of a random convex combination of two of these waveforms sampled at the integers with noise added. More specifically, the measurement vectors are 21 dimensional: $\boldsymbol{x} = (x_1, \cdots, x_{21})$. To generate a class 1 vector $\boldsymbol{x}$, independently generate a uniform random number $u$ and 21 random numbers $\epsilon_1, \cdots, \epsilon_{21}$ normally distributed with mean zero and variance 1. Then, set

$$x_m = uw_1(m) + (1-u)w_2(m) + \epsilon_m, \qquad m = 1, \cdots, 21.$$

To generate a class 2 vector, repeat the preceding and set

$$x_m = uw_1(m) + (1-u)w_3(m) + \epsilon_m, \qquad m = 1, \cdots, 21.$$

Class 3 vectors are generated by

$$x_m = uw_2(m) + (1-u)w_3(m) + \epsilon_m, \qquad m = 1, \cdots, 21.$$

Three hundred measurement vectors were generated using prior probabilities of $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, so there were approximately 100 per class.

In one set of data generated as above, the CART tree classifier had a test set error rate of 28%. Using linear combinations in the tree construction lowers the test set error rate to 20%. Linear discriminant analysis with stepwise entry of variables gave a test set error rate of 26%.

Ten data sets were generated using the above mechanism, together with ten test sets of size 3000. When each training set hinges were added and refitted. The model selected (i.e., number of hinges fitted) was that with minimum test set error.

For the stochastic structure of this data, the Bayes rule can be computed and applied to the test sets. The averages over the 10 repetitions are:

| | |
|---|---|
| number of hinges | 3.6 |
| training set misclassification rate (%) | 9.5 |
| test set misclassification rate (%) | 18.1 |
| Bayes rule misclassification rate (%) | 13.6 |
| cpu seconds (RS 6000) | 39.7. |

The hinges procedure improves on the tree classifier using linear combination splits. But the error rate is still somewhat above the Bayes rate. We revisit this example in the next section.

### C. Stepwise Addition of Variables and More Examples

For high-dimensional problems a variable selection method is imperative. The method used in classification is an extension of the regression method. Suppose that hinge functions $h_1, \cdots, h_{K-1}$ have been selected and residuals $\{r_{jn}\}$ computed. What is wanted is a hinge $h_K$ and $\{\gamma_j\}$ to minimize (5.2).

The stepwise procedure first finds the single variable hinge (say on $x_1$) and coefficients $\{\gamma_j\}$ that minimize (5.2). Then it finds the two variable hinge based on $(x_1, x_m)$, $m \neq 1$, and coefficients $\{\gamma_j\}$ that minimize (5.2). Variables are added until the $\mathrm{PE}_{\mathrm{GCV}}$ criterion derived in Appendix III becomes minimum. Then refitting is carried out among all $K$ hinge functions. The number of hinge functions fitted is determined by test set or cross validation.

### D. The Wave Form Example Revisited

The simulated data of section 5.2 was rerun using the stepwise procedure. The results (averaged over the 10 repetitions) were:

| | |
|---|---|
| number of hinges | 3.7 |
| number of variables per hinge | 9.3 |
| training set misclassification rate (%) | 10.5 |
| test set misclassification rate (%) | 17.3 |
| Bayes rule misclassification rate (%) | 13.6 |
| cpu seconds | 7.3. |

Note that the accuracy increases by 0.8% while the compute time has been cut from 40 cpu seconds to 7 cpu seconds. The decrease of 0.8% seems insignificant, but another way to look at it is that the all variable procedure is 4.5% above the Bayes rate while the stepwise procedure is 3.7% above.

### E. High-Dimensional Simulated Data

To test the stepwise entry method on a larger problem, we generated 10 class, 61 dimensional data with 1000 cases. In structure, it is an extension of the 3 class wave form data.

The wave form function $w(x) = 10 - \min(10, |x|)$, has a peak at zero of height 10, and is zero outside of $[-10, 10]$. Consider five functions

$$w_k(m) = w(m - 10k - 1), \qquad k = 1, \cdots, 5,$$

where $m = 1, \cdots, 61$. These functions are centered at the points 11, 21, 31, 41, 51. Let $S_1, \cdots, S_{10}$ consist of all subsets of size three of $\{1, \cdots, 5\}$, i.e., $S_1 = \{3, 4, 5\}$ $S_2 = \{2, 4, 5\}, \cdots, S_{10} = \{1, 2, 3\}$.

The data is generated as follows: For $n = 1$ to 1000, the class $j \in 1, \cdots, 10$ is selected with probability 0.1 of each choice. Three uniform random numbers $u_1, u_2, u_3$ are generated. For the lowest index $k_1$ in $S_j$, $q(k_1)$ is defined as $u_1/(u_1 + u_2 + u_3)$; for the second lowest index $k_2$ in $S_j$, $q(k_2) = u_2/(u_1 + u_2 + u_3)$ and for the third index $q(k_3) = u_3/(u_1 + u_2 + u_3)$. For $k \notin S_j$, $q(k) = 0$. Then, for $m = 1$ to 61,

$$x(m, n) = \sum_k q(k) w(m, k) + \epsilon_m,$$

where the $\{\epsilon_m\}$ are independent unit normals.

Two areas were investigated with this data. First was the question of how accurately could classification be done using linear methods only. The second was how accuracy was effected by changing the constant in the criterion which governs the number of variables entered into a hinge.

To explore the accuracy of linear methods, a stepwise entry of variables into a linear classifier was employed. Let $S \subset \{0, \cdots, 61\}$ be the subset of variables already entered, with $|S| = K$. Then the $(K+1)$st variable entered is that $x_k$, $k \notin S$ that minimizes

$$\min_{\{c_{jm}, c_{j,k}\}} \sum_{j,n} \left( y_{n,j} - \sum_{m \in S} c_{j,m} x_{mn} - c_{j,k} x_{kn} \right)^2.$$

For each $K$ a 3000 case test set was used to estimate the classification error. The minimum test set misclassification rate was 37.3% occurring when 12 variables were entered.

The second question is this: the $\mathrm{PE}_{\mathrm{GCV}}$ criterion given in appendix II keeps adding variables to a hinge as long as the $\mathrm{PE}_{\mathrm{GCV}}$ value is decreasing. Using $c = 1.5$, the decreases are quite small for a long string of entries. In this case, is the program using up degrees of freedom for small gain, and could more accuracy be accomplished by going on the next hinge? To investigate this, we ran the program using $c = 1.5$, 2.0, 3.0, 6.0. Table VI gives the results for the number of hinges giving the minimum test set error. The accuracy results are reasonably insensitive to the number of variables per hinge—equivalently, to the value of $c$ between 1.5 and 6.0. The evidence of this example argues for a value of $c$ between 2.0 and 3.0. But in the smaller waveform example increasing $c$ from 1.5 to 2.0 increases the test set error to 17.7%.

The nonlinear hinge procedure improves considerably on the linear error rate. Unfortunately, the Bayes procedure is difficult to calculate, so there is no way of telling how much better can be done.

TABLE VI
SUMMARY OF RUNS ON EXTENDED WAVEFORM DATA

| | Value of constant | | | |
|---|---|---|---|---|
| | 1.5 | 2.0 | 3.0 | 6.0 |
| Number of Hinges | 4 | 4 | 4 | 4 |
| Average No. Variables/Hinge | 25.5 | 16.8 | 11.5 | 6.5 |
| Training Set Error (%) | 14.0 | 13.9 | 15.5 | 17.3 |
| Test Set Error (%) | 19.2 | 18.6 | 18.6 | 19.8 |
| CPU seconds | 135.4 | 84.0 | 70.1 | 40.4 |

## VI. CONCLUSION

Fitting hinge functions to data can give good predictive results in regression, classification and noiseless function approximation. Using stepwise entry of variables into the hinges, accurate predictors in fairly large problems can be derived in short compute times. These preliminary results are encouraging, but there are important questions that need further work.

1) Choice of a good initial starting hinge is important. The current strategy uses $M$ different starting hinges in $M$ dimensions and takes the best result. Is it possible to find a single starting hinge that will provide uniformly good results?
2) The stepwise entry method used is crude. A more refined procedure, for example, is to enter two variables at a time and then delete from the set of entered variables that single variable whose deletion causes the least rise in RSS. Can more sophisticated stepwise entry procedures significantly improve accuracy?
3) In stepwise entry, the variables in the first hinge sometimes appear to be selected almost by happenstance, as the algorithm is hunting around to get some purchase. Is there a way to improve the variable selection in the first hinge?
4) The hinge algorithm is not guaranteed to find the global minimum at $\|f - \sum_1^K h_k\|$ over $h_1, \cdots, h_K$. If $f(\boldsymbol{x})$ is badly behaved, the hinge algorithm may converge to a poor fit. Is there a way to consistently get near the global minimum?

This methodology will be explored in the binary tree context, and using a related iterative approach, research into fitting by sums and products of ramp functions is being carried out in collaboration with Jerome Friedman. (A ramp function is a continuous function $y = r(\boldsymbol{x})$ formed by the intersection of a hyperplane with two planes of constant height).

## APPENDIX A
## PROOFS

*Proof of Theorem 1:* Without loss of generality (w.l.o.g.), we can assume that $\Gamma = I$ and $\|\Delta^*\| = 1$. For any $\Delta^{(0)}$, let $S_+ = \{\boldsymbol{x}; \Delta^{(0)} \cdot \boldsymbol{x} \geq 0\}$, $S_-$ the complement space, and define $\Gamma_+, \Gamma_-$ as previously. Using an orthogonal transformation on $x_0, x_1, \cdots, x_m$ we can reduce $\Gamma_+, \Gamma_-$ to diagonal form

$D_+$, $D_-$, where $D_+ = I - D_-$. By (2.1),

$$\Delta^* - \Delta^{(1)} = D_+^{-1}\Gamma_{+-}\Delta^* + D_-^{-1}\Gamma_{-+}\Delta^*.$$

Let $s = \Delta^* \cdot \boldsymbol{x}$, and $\Delta^{(0)} \cdot \boldsymbol{x} = \alpha s + \overline{\alpha}t$, $E(st) = 0$, $E(t^2) = E(s^2) = 1$. Then, $(\delta^{(0)})^2 = \|\Delta^{(0)} - \Delta^*\|^2 = (1-\alpha)^2 + \overline{\alpha}^2$. For $\mu = \Delta^* - \Delta^{(1)}$,

$$\begin{aligned}\mu \cdot (\Delta^* - \Delta^{(1)}) &= \mu D_+^{-1}\Gamma_{+-}\Delta^* + \mu D_-^{-1}\Gamma_{-+}\Delta^* \\ &= \mu^+\Gamma_{+-}\Delta^* + \mu^-\Gamma_{-+}\Delta^* \\ &= \int_{S_{+-}} (\mu^+ \cdot \boldsymbol{x}) s\, dP + \int_{S_{-+}} (\mu^- \cdot \boldsymbol{x}) s dP.\end{aligned}$$

Take $\mu^+ = \|\mu^+\| e^+$, $\|e^+\| = 1$, and similarly for $\mu^-$ and note that $\|\mu^+\| \le \|\mu\|/\lambda_+$, $\|\mu^-\| \le \|\mu\|/\lambda_-$ where $\lambda_+$, $\lambda_-$ are the minimum eigenvalues of $\Gamma_+$, $\Gamma_-$. Thus,

$$\begin{aligned}\delta^{(1)} = \|\Delta^* - \Delta^{(1)}\| \le \frac{1}{\lambda_+}\int_{S_{+-}} |v^+ s|\, dP \\ + \frac{1}{\lambda_-}\int_{S_{-+}} |v^- s|\, dP\end{aligned}$$

where $v^+ = e^+ \cdot \boldsymbol{x}$, $v^- = e^- \cdot \boldsymbol{x}$. Assume that $\delta^{(0)} < 1$, then $\alpha > 0$. $S_{-+}$ is the set $s \ge 0$, $\alpha s + \overline{\alpha}t < 0$, which combines into $0 \le s < -(\overline{\alpha}/\alpha)t$. $S_{+-}$ is the set $0 > s \ge -(\tilde{\alpha}/\alpha)t$. Letting $\gamma = |\overline{\alpha}/\alpha|$,

$$\delta^{(1)} \le \frac{\gamma}{\lambda_+}\int_{S_{+-}} |v^+ t|\, dP + \frac{\gamma}{\lambda_-}\int_{S_{-+}} |v^- t|\, dP.$$

Define

$$\theta(\gamma) = \sup \max_{v^+, v^-, t}\left[\int_{S_{+-}} |v^+ t|\, dP, \int_{S_{-+}} |v^- t|\, dP\right],$$

then $\theta(\gamma) \le 1$, and $\theta(\gamma) = O(\gamma^{2/5})$ as $\gamma \to 0$. To verify these statements, note that

$$\int_T |vt|\, dP \le \sqrt{Ev^2 Et^2} = 1,$$

and

$$\int_T |vt|\, dP \le (P(T))^{1/2}(Ev^4 Et^4)^{1/4}.$$

Now $\max(P(S_{+-}), P(S_{-+})) \le P(|s| \le \gamma|t|)$, and the latter is dominated by $p(\gamma b) + c/b^4$. Taking $b = \gamma^{-1/5}$ makes this bound equal to

$$q(\gamma) = p(\gamma^{4/5}) + c\gamma^{4/5}.$$

Now,

$$\begin{aligned}P(\Delta^{(0)} \cdot \boldsymbol{x} \ge 0) &= P(\Delta^* \cdot \boldsymbol{x} \ge (\Delta^* - \Delta^{(0)}) \cdot \boldsymbol{x}) \\ &\ge P(\Delta^* \cdot \boldsymbol{x} \ge 0) \\ &\quad - P(|\Delta^* \cdot \boldsymbol{x}| \le |(\Delta^* - \Delta^{(0)}) \cdot \boldsymbol{x}|).\end{aligned}$$

The last term is bounded by

$$P(|\Delta^* \cdot \boldsymbol{x}| \le \alpha) + \frac{c}{\alpha^4}\delta^{(0)^4}.$$

With $\alpha = (\delta^{(0)})^{4/5}$,

$$P(\Delta^{(0)} \cdot \boldsymbol{x} \ge 0) \ge P(\Delta^* \cdot \boldsymbol{x} \ge 0) - q(\delta^{(0)}).$$

Let $c^* = P(\Delta^* \cdot \boldsymbol{x} > 0)$, then,

$$\delta^{(1)} \le \gamma\theta(\gamma)\left[\frac{1}{\lambda(c^* - q(\delta^{(0)}))} + \frac{1}{\lambda(1 - c^* - q(\delta^{(0)}))}\right].$$

Noting that $\gamma \le \delta^{(0)}/\sqrt{1 - (\delta^{(0)})^2}$ gives the result that

$$\delta^{(1)} \le \delta^{(0)}\phi(\delta^{(0)}),$$

where $\phi(\delta^{(0)}) = O((\delta^{(0)})^{2/5})$. □

*Proof of Theorem 2:* W.l.o.g., we can take $x_1, \cdots, x_M$ i.i.d. $N(0, 1)$ such that $(\Delta^* \cdot \boldsymbol{x}) = x_1$. For any $\Delta^{(0)}$, let $s = (\Delta^{(0)} \cdot \boldsymbol{x})$. Then,

$$\Gamma_+ = \Gamma_- = I/2$$

and

$$\begin{aligned}\Delta^{(1)}(m) = \Delta^*(m) - 2\{E(x_m x_1;\ x_1 \ge 0,\ s < 0) \\ + E(x_m x_1;\ x_1 < 0,\ s > 0)\}.\end{aligned}$$

Let $\|\Delta^{(0)}\|_1 = \sqrt{\|\Delta^{(0)}\|^2 - \Delta^{(0)}(1)^2}$, and $t^{(0)} = \Delta^{(0)}(1)/\|\Delta^{(0)}\|_1$. By simple manipulations,

$$\begin{aligned}\Delta^{(1)}(1) &= \theta(t^{(0)}) \\ \Delta^{(1)}(m) &= \frac{\Delta^{(0)}(m)}{\|\Delta^{(0)}\|_1} X(t^{(0)}), \qquad m \ge 2,\end{aligned}$$

where

$$\theta(t) = 1 - 4\int_0^\infty \boldsymbol{x}^2 \Phi(-tx) f(x)\, dx$$

$$X(t) = \frac{2}{\pi} \cdot \frac{1}{1 + t^2},$$

$f(x)$ is the standard normal density and $\Phi(x) = \int_{-\infty}^x f(y)\, dy$. Therefore,

$$t^{(1)} = \phi(t^{(0)}),$$

where

$$\phi(t) = \frac{\pi}{2}(1 + t^2)\theta(t).$$

Now, $\theta'(t) = (4/\pi)(1 + t^2)^{-2}$, so for $t > 0$,

$$\theta(t) > \frac{4}{\pi t}\int_0^t \frac{\xi\, d\xi}{(1 + \xi^2)^2} = \frac{2t}{\pi(1 + t^2)}.$$

In consequence, $\phi(t) > t$, all $t > 0$, and $t^{(k)} \to \infty$ unless $t^{(0)} = 0$, so $\Delta^{(k)}(1) \to 1$ and $\Delta^{(k)}(m) \to 0$, $m \ge 1$. □

*Proof of Theorem 3:* The key is a lemma which we reproduce from Barron [1].

*Lemma:* If $f$ is in the closure of the convex hull of a set $G$ in a Hilbert space, with $\|g\| \le b$, all $g \in G$, then for every $K \ge 1$ and every $c' > b^2 - \|f\|^2$ there is an $f_K$ in the convex hull of $K$ points in $G$ such that $\|f - f_K\|^2 \le c'/K$.

Denote by $h^+(x)$, the $x^+$ function (i.e., $h^+(x) = x$, $x \ge 0$, else $= 0$). Every hinge function is of the form

$$\beta_0 + \beta \cdot \boldsymbol{x} + h^+(\Delta_0 + \Delta \cdot \boldsymbol{x}).$$

The idea of the proof is to show that

$$f(\boldsymbol{x}) - \boldsymbol{x} \cdot \nabla f(0) - f(0)$$

is in the closure of the convex hull of a set of functions $\{h^+(\Delta_0 + \Delta \cdot \boldsymbol{x})\}$ with such that each function in the set has norm bounded by $c(2R)^2$.

Begin by noting that

$$\begin{aligned} g(\boldsymbol{x}) &= f(\boldsymbol{x}) - \boldsymbol{x} \cdot \nabla f(0) - f(0) \\ &= Rl \int (e^{iw \cdot \boldsymbol{x}} - iw \cdot \boldsymbol{x} - 1)\tilde{f}(w)\, dw \\ &= Rl \int (e^{iw \cdot \boldsymbol{x}} - iw \cdot \boldsymbol{x} - 1)e^{i\theta(w)}|\tilde{f}(w)|\, dw \\ &= \int [\cos(w \cdot \boldsymbol{x} + \theta(w)) + w \cdot \boldsymbol{x} \sin\theta(w) \\ &\quad - \cos\theta(w)]|\tilde{f}(w)|\, dw. \end{aligned}$$

Write $Q(dw) = \|w\|^2|\tilde{f}(w)|/c$, so

$$\begin{aligned} g(\boldsymbol{x}) = c \int [&\cos(w \cdot \boldsymbol{x} + \theta(w)) + w \cdot \boldsymbol{x} \sin\theta(w) \\ &- \cos\theta(w)] \frac{Q(dw)}{\|\boldsymbol{w}\|^2} \end{aligned}$$

and $g(\boldsymbol{x})$ is in the closure of the convex hull of the functions

$$s_w(\boldsymbol{x}) = \frac{c[\cos(w \cdot \boldsymbol{x} + \theta(w)) + w \cdot \boldsymbol{x} \sin\theta - \cos\theta]}{\|w\|^2}.$$

Let $w = \|w\|\alpha$, $b = \theta(w)$, $y = \alpha \cdot \boldsymbol{x}$, and

$$\phi(y) = \frac{c[\cos(\|w\|y + b) - \|w\|y \sin b - \cos b]}{\|w\|^2},$$

so $s_w(\boldsymbol{x}) = \phi(y)$, $\|\alpha\| = 1$, $y \in [-R, R]$, $\phi(0) = 0$, and $\phi''(y)$ is continuous on $[-R, R]$. Consider fitting $\phi(y)$ at the points $j/J$, $j = 0, \cdots, \text{int}(RJ)$ with a function of the form

$$s_J(y) = \sum_0^{\text{int}(RJ)} \alpha_i h^+(y - i/J).$$

Then, $\sum_1^{j-1} \alpha_i = J[\phi(j/J) - \phi(j - 1/J)]$, and $\alpha_i$ is $J$ times the second difference of the sequence $\phi(j/J)$ evaluated at $j = i$. Thus, as $J \to \infty$,

$$\sum_i |\alpha_i| \to \int_0^R |\phi''(y)|\, dy.$$

A similar argument holds for approximating $\phi(y)$ on $[-R, 0]$, with sums of functions of the form $h^+(-y - i/J)$. Thus, with

$$\int_{-R}^{+R} |\phi''(y)|\, dy \le 2cR,$$

$\phi(y)$ is in the closure of the convex hull of the set of functions $\{h^+(\gamma(y - \tau))\}$ with $|\gamma| \le 2cR$ and $\tau \in [-R, R]$. This implies that $g(\boldsymbol{x})$ is in the closure of the convex hull of the set of functions $\{h^+(\gamma(\alpha \cdot \boldsymbol{x} - \tau))\}$ with $\|\alpha\| = 1$. Since

$$\begin{aligned} &\int [h^+(\gamma(\alpha \cdot \boldsymbol{x} - \tau))]^2\, dP \\ &\le \gamma^2 \int (\alpha \cdot \boldsymbol{x} - \tau)^2\, dP \le (2R)^2\gamma^2, \end{aligned}$$

using the lemma completes the proof. □

## Appendix B
## Computational Aspects of Hinge Fitting

There are devices used in the hinge programs both to improve accuracy and improve computing speed. We mention these to assist other researchers in building their own hinge programs.

The basic building block is the hinge finding algorithm. In this, two devices are noted: 1) if, in the iterations, the hinge gets too close to the edge of the data, then it is pushed back into the data. More precisely, if the number of points on one side of the hinge (say $\Delta x \le 0$) falls a threshold value $NH$, then $\Delta(0)$ is readjusted so that $\#\{\boldsymbol{x}_n;\ \Delta \cdot \boldsymbol{x}_n \le 0\} = NH$.

The major computational burden in the algorithm is usually in the recomputation of $S_+ = (X^tX)_+$, $S_-$ after a new hinge has been selected. To reduce this, the sign of the new hinge is taken so that the number of points in $S_+$ is $\le \text{int}(N/2)$, $S_+$ is updated only for those points that have changed sides, and $S_-$ set equal to $S - S_+$.

In the refitting algorithm, the last hinge $\Delta_k$ used in a hinge function $h_k$ is stored. Any refitting of $h_k$ starts from $\Delta_k$ and the new hinge is then stored in place of $\Delta_k$. In the refitting algorithm, the best performance has been gotten by using one iteration of the hinge finding algorithm per hinge function per cycle. Often, there are many cycles until the reduction in RSS levels off.

In stepwise variable selection the problem was to construct a computationally feasible procedure when dimensionality is high—say over 50 variables. The device used is this: suppose variables $x_1, \cdots, x_L$ have been entered into the current hinge. What is stored is an indicator vector $id(n)$, which tells which side of the current hinge each $\boldsymbol{x}_n$ is on, the matrices $S_+^{-1}$, $S_-^{-1}$, and the values of $x_l \cdot x_m$ for all $l \le L$ and $m > L$. For a variable $x_m$, $m > L$, the indicator vector id is used to compute $(x_l \cdot x_m)_+$ for all $l \le L$, $(x_m \cdot x_m)_+$, $(x_m \cdot y)_+$ and the corresponding $(\cdot)_-$ values. Then, a fast update formula uses these values to produce the inverses $\tilde{S}_+^{-1}$, $\tilde{S}_-^{-1}$, where the tilde indicates that the $m$th variable values are included in the $\tilde{S}_+$, $\tilde{S}_-$ matrices.

Then, hyperplanes based on $(x_1, \cdots, x_L, x_m)$ are fitted on either side of the current hinge, the new hinge function computed along with the new residual-sum-of squares $\text{RSS}_m$. The variable added is that with minimum $\text{RSS}_m$. In actuality, computing $\text{RSS}_m$ requires only a partial update of $S_+^{-1}$, $S_-^{-1}$ and after $m$ is selected, then the full update to $\tilde{S}_+^{-1}$, $\tilde{S}_-^{-1}$ is done. The final step is to run the hinge finding algorithm to convergence using $(x_1, \cdots, x_L, x_m)$ and store as starting values for the next round the final $\tilde{S}_+^{-1}$, $\tilde{S}_-^{-1}$ and $\tilde{id}$.

The stepwise forward procedure in classification adds an extra step. After the update to $\tilde{S}_+$, $\tilde{S}_-$ and $\tilde{Z}_+ = (x\tilde{y})_+$, $\tilde{Z}_- = (x\tilde{y})_-$, form the updated $H$ matrix (selection V-B), and solve the $J \times J$ eigenvalue problem to find the new hinge and RSS value. In the $L$ variable case, either a $J \times J$ or $L \times L$ eigenvalue provides the new hinge. If $J$ is large, it will be faster to solve the sequence of $L \times L$ problems.

## Appendix C
## Cross-Validation Measures

An important question in the use of hinges is: how many hinges to use? If stepwise entry of variables is used, another question is how many variables to enter into the current hinge. If large test sets are available, they can be used to answer these questions. If not, then we use cross-validation measures as our basic tool.

### *Number of Hinges Used in Regression*

One approach is to use 10-fold cross-validation (see Breiman *et al.* [5]). However, this increases the computations by a factor of ten. Another approach is to use cross-validation measures based on the leave-one-out method. That is, the case $\boldsymbol{x}_n$ is left out of the fitting, coefficients derived and then $\boldsymbol{x}_n$ used as a single case test set. This is repeated for each case in the data and then the mean taken of the sums-of-squares of test set errors. The resulting measure is called the cross-validated prediction error, or $\mathrm{PE_{CV}}$. In ordinary regression, it is not difficult to show that

$$\mathrm{PE_{CV}} = \frac{1}{N}\sum_1^N (r_n/1 - h_n)^2,$$

where $r_n$ is the usual residual and $h_n$ is the $n$th diagonal element of the matrix $H_{nn'} = \boldsymbol{x}_n^t S^{-1} \boldsymbol{x}_{n'}$ where $S^{-1}$ is the inverse of the matrix $\sum_n x_{mn} x_{m'n}$. The trace of $H$ is the number of variables $M$ in the equation. If $h_n$ is replaced by its average $\sum_n h_n/N$ in the $\mathrm{PE_{CV}}$ expression, the resulting approximation is called the generalized cross-validation estimation of the prediction error and is given by

$$\mathrm{PE_{GCV}} = \mathrm{MRSS}/(1 - M/N)^2.$$

In fitting hinge functions, recall that the fit is the sum of a linear function and a function of the type $h^+(\boldsymbol{\Delta} \cdot \boldsymbol{x})$. That is, RSS is the minimum of

$$\sum_n \left( y_n - \beta \cdot \boldsymbol{x}_n - \sum_{k=1}^K h^+(\boldsymbol{\Delta}_k \cdot \boldsymbol{x}_n) \right)^2. \tag{A.3.1}$$

Denote by $\beta$, $\Delta_k$ the minimizers of (A.3.1), and $\{r_n\}$ the residuals. In the neighborhood of $\beta$, $\boldsymbol{\Delta}_k$, say $\beta' = \beta + \epsilon$, $\boldsymbol{\Delta}_k' = \boldsymbol{\Delta}_k + \delta_k$, (A.3.1) is approximated by

$$\sum_n \left( r_n - \epsilon \cdot \boldsymbol{x}_n - \sum_{k=1}^K \delta_k \cdot \boldsymbol{x}_n I(\boldsymbol{\Delta}_k \cdot \boldsymbol{x}_n \geq 0) \right)^2, \tag{A.3.2}$$

where $I(\cdot)$ is the indicator function.

Define the $(M+1)(K+1)$ dimensional variable $\xi$ by

$$\xi = \{\boldsymbol{x}, \boldsymbol{x} I(\Delta_1 \cdot \boldsymbol{x} \geq 0), \cdots, \boldsymbol{x} I(\Delta_K \cdot \boldsymbol{x} \geq 0\}.$$

Then, letting $\alpha = (\epsilon, \delta_1, \cdots, \delta_K)$, (A.3.2) has the form

$$\sum_n (r_n - \alpha \cdot \xi_n)^2. \tag{A.3.3}$$

If the case numbered $n'$ is deleted and the deleted sum-of-squares in (A.3.1) minimized, the resulting $\beta'$, $\boldsymbol{\Delta}_k'$ will be close to $\beta$, $\Delta_k$. Therefore, the deletion and cross-validation measures for the nonlinear (A.3.1) can be approximated by the same measures for the linearized (A.3.3). Thus, to first order, the $\mathrm{PE_{GCV}}$ for the hinge fitting is given by

$$\mathrm{PE_{GCV}} = \mathrm{MRSS}/(1 - N_p/N)^2, \tag{A.3.4}$$

where $N_p = (K+1)(M+1)$.

Because of the replacement of $h_n$ by $\overline{h}$, the expression for $\mathrm{PE_{GCV}}$ given by (A.3.4) is usually biased low. The simplest way to correct this bias is to put

$$\mathrm{PE_{GCV}} = \mathrm{MRSS}(1 - cN_p/N)^2,$$

where $c \in [1, 3]$. We have used $c = 1.5$ throughout with fairly accurate results, although the choice of the number of hinges selected is fairly insensitive to the choice of $c$.

### *Number of Variables Used in a Hinge (Regression)*

Suppose that some hinges have already been fitted, the current residuals are $\{r_n\}$ and stepwise entry of variables into the next hinge function is being carried out to fit the $\{r_n\}$. Consider fitting with a single hinge function of $M_0$ variables. Then, from the previous derivation leading to (A.3.4), we get

$$\mathrm{PE_{GCV}} = \mathrm{MRSS}/(1 - N_p/N)^2,$$

where $N_p = 2(M_0 + 1)$.

In the stepwise selection of variables context, there is an additional reason why this expression is biased low. In brief, the cross-validation measure previously used does not cross-validate the variable selection process (see Breiman [3], Breiman and Spector [4]). Again, the estimate

$$\mathrm{PE_{GCV}} = \mathrm{MRSS}/(1 - cN_p/N)^2$$

is used ($c = 1.5$ throughout), and variables are added until this expression starts to increase.

### *Number of Hinges Used in Classification*

An expression can be developed for the leave-one-out misclassification rate, but it is complex and difficult to evaluate. For $K$ hinges and $M$ variables it requires the inversion of a matrix of order $K(M+1) \times K(M+1)$. For 99 variables with 5 hinges, this is a $500 \times 500$ matrix. If forward stepwise entry of variables is used, the matrix is sparse, and efficient methods may be found to invert it. In large problems a test set provides a rapid method for determining how many hinges to use. If the sample size is not large enough for a test set, we recommend 10-fold cross-validation.

*Number of Variables Used in a Hinge (Classification)*

Unless the test set is large, using it to both determine the number of hinges and the number of variables in each hinge may be overfitting the test set. The procedure outlined below has proven a satisfactory alternative in the examples we have run. It assumes that the "true misclassification rate" is decreasing as long as the "true regression error" is decreasing.

Suppose there are currently $M_0$ variables entered into the $K$th hinge, and denote the residuals after the $(K-1)$st hinge was fitted by $\{r_{jn}\}$. The $\text{PE}_{\text{GCV}}$ when $M_0$ variables are used will be computed and variables added until the $\text{PE}_{\text{GCV}}$ stops decreasing.

Let the coefficients $\{c_j\}$ and hinge function $h(\boldsymbol{x}) = \beta \cdot \boldsymbol{x} + h^+(\boldsymbol{\Delta} \cdot \boldsymbol{x})$ based on $M_0$ variables minimize

$$\sum_{j,\,n}(r_{jn} - c_j h(\boldsymbol{x}_n))^2. \tag{A.3.5}$$

Denote $z_{jn} = r_{jn} - c_j h(\boldsymbol{x}_n)$. Then $\boldsymbol{z}_j \perp \boldsymbol{h}$, $j = 1, \cdots, J$; and if $\mu = \sum_j c_j \boldsymbol{z}_j$, then $\mu \perp \boldsymbol{x}_m$, $m = 0, \cdots, M_0$. Further, $\mu \perp \boldsymbol{t}_m$, $m = 0, \cdots, M_0$ where $\boldsymbol{t}_m = \boldsymbol{x}_m I(\boldsymbol{\Delta} \cdot \boldsymbol{x} > 0)$. Let $\beta' = \beta + \epsilon$, $\Delta' = \Delta + \delta$, and $c_j' = c_j + \sigma_j$. Then, the linearized version of (A.3.5) is

$$\sum_{j,\,n}(z_{jn} - \sigma_j h(\boldsymbol{x}_n) - c_j(\epsilon \cdot \boldsymbol{x}_n + \delta \cdot \boldsymbol{t}_n))^2. \tag{A.3.6}$$

Define the $2(M_0+1)$ dimensional variable $\xi_n = (\boldsymbol{x}_n, \boldsymbol{t}_n)$, and $\boldsymbol{s} = (\epsilon, \delta)$ so that A.3.6 becomes

$$\sum_{j,\,n}(z_{jn} - \sigma_j h(\boldsymbol{x}_n) - c_j \boldsymbol{s} \cdot \xi_n)^2. \tag{A.3.7}$$

If the $n'$ case is deleted, then the RSS is

$$\sum_{j,\,n}(z_{jn} - \sigma_j h(x_n) - c_j \boldsymbol{s} \cdot \xi_n)^2 - \sum_{j}(z_{jn'} - \sigma_j h(\boldsymbol{x}_{n'}) - c_j \boldsymbol{s} \cdot \xi_{n'})^2. \tag{A.3.8}$$

Because of the orthogonality relations, the first term is

$$\sum_{j,\,n} z_{jn}^2 + \sum_{j,\,n}(\sigma_j h(x_n) + c_j \boldsymbol{s} \cdot \xi_n)^2.$$

If the second term in (A.3.8) is expanded only to first-order terms in $\sigma_j$, $\boldsymbol{s}$, and we assume $\sum c_j^2 = 1$, then partial derivatives with respect to $\sigma_j$ and $\boldsymbol{s}$ give the equations

(A) $\sigma_j \|h\|^2 + c_j \boldsymbol{s}(\xi, h) = -z_{jn'} h_{n'}$

(B) $(\sum_j \sigma_j c_j)(\xi, h) + S\boldsymbol{s} = -u_{n'} \xi_{n'}$

where $u_{n'} = \sum_j c_j z_{jn'}$, and $S_{mm'} = (\xi_m, \xi_{m'})$.

Equation (A) gives $\sigma_j$ in terms of $\boldsymbol{s}$. The solution for $\boldsymbol{s}$ (to first order) is as follows: define a matrix $A$ by

$$A_{mm'} = S_{mm'} - \frac{(\xi_m, h)(\xi_{m'} h)}{\|h\|^2}$$

and a vector $\boldsymbol{v}_n$ by

$$\boldsymbol{v}_n = \left(\xi_n - \frac{(\xi, h)h_n}{\|h\|^2}\right).$$

Then,

$$\boldsymbol{s} = -u_{n'} A^{-1} \boldsymbol{v}_{n'},$$

and

$$\text{PE}_{\text{CV}} \cong \sum_{j,\,n}(z_{jn} - \sigma_{jn} h(x_n) - c_j \boldsymbol{s}_n \cdot \xi_n)^2, \tag{A.3.9}$$

where the subscripts $n$ on $\sigma_{jn}$, $\boldsymbol{s}_n$ indicate that these are the values of $\sigma_j$, $\boldsymbol{s}$ gotten by minimizing the RSS with the $n$th case deleted. Let $\text{RSS} = \sum_j (\boldsymbol{z}_j, \boldsymbol{z}_j)$. To terms of order RSS/N, A.3.9 equals

$$\text{RSS} + \sum_n u_n^2 \boldsymbol{v}_n^t A^{-1} \boldsymbol{v}_n.$$

To approximate the second term, we replace $u_n^2$ by $\overline{u}^2$. Noting that $\sum_n \boldsymbol{v}_n^t \boldsymbol{v}_n = A$, the approximation equals $2(M_0+1)\overline{u}^2$. Therefore,

$$\text{PE}_{\text{GCV}} \cong \text{MRSS} + 2(M_0+1)\overline{u}^2/N.$$

Analysis of second-order terms shows that a slightly more accurate value is

$$\text{PE}_{\text{GCV}} \cong \text{MRSS} - \overline{u}^2 + \overline{u}^2/(1 - N_p/N)^2$$

with $N_p = 2(M_0+1)$. As previously, we correct this for downward bias by using

$$\text{PE}_{\text{GCV}} = \text{MRSS} - \overline{u}^2 + \overline{u}^2/(1 - cN_p/N)^2.$$

In two examples, the evidence indicates that the more selection is going on (i.e., the higher the dimensionality) the larger $c$ should be. In 21 dimensions, $c = 1.5$ was better than $c = 2$. In 61 dimensions, the best value of $c$ is between 2.0 and 3.0. But in both cases, the test set error changes only slightly over a wide range of $c$.

## REFERENCES


[1] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," Tech. Rep. #58, Dept. of Statist., Univ. of Illinois at Urbana-Champaign, 1991.

[2] L. Breiman, "The $\pi$ method for estimating multivariate functions from noisy data," (with discussion), *Technometrics*, vol. 33, no. 2, pp. 125–160, 1991.

[3] ——, "The little bootstrap and other methods for dimensionality selection in regression: $X$-fixed prediction error," Tech. Rep. No. 169, Statist. Dept. Univ. of California, Berkeley, CA, 1990.

[4] L. Breiman and P. Spector, "Submodel selection and evaluation in regression: The $X$-random case," Tech. Rep. No. 197, Statist. Dept. Univ. of California, Berkeley, CA, 1990.

[5] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Wadsworth Inc, 1984.

[6] L. Breiman and J. Friedman, "Estimating optimal transformations in regression and correlation," (with discussion), *J. Amer. Statist. Assoc.*, vol. 80, pp. 580–619, 1985.

[7] L. Breiman and R. Ihaka, "Nonlinear discriminant analysis via scaling and ACE," Tech. Rep. #40, Statist. Dept., Univ. of California, Berkeley, 1987.

[8] J. Friedman, "Multivariate adaptive regression splines," (with discussion), *Ann. Statist.*, vol. 19, no. 1, pp. 1–141, 1991.

[9] J. Friedman and W. Stuetzle, "Projection pursuit regression," *J. Amer. Statist. Assoc.*, vol. 76, pp. 817–823, 1981.

[10] T. Hastie and R. Tibshirani, *Generalized Additive Models*. London: Chapman and Hall.

[11] L. Jones, "A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training," Tech. Rep. No. 16, Dept. of Math., Univ. of Lowell, Lowell, MA, 1990.

[12] C. Lee and R. Lippmann, "Practical characteristics of neural network and conventional pattern classifiers on artificial and speech problems," *NIPS 89*, Denver, CO, Nov. 1989.
[13] R. Lippmann, "Pattern classification using neural networks," *IEEE Commun. Mag.*, vol. 11, pp. 47–64, 1989.
[14] W. Meisel and C. Collins, "Repro-modeling: An approach to efficient model utilization and interpretation," *IEEE Trans. Syst. Man Cybern.*, vol. 3, pp. 349–358, July–Aug. 1973.
[15] G. Wahba, "Spline functions for observational data," presented at *SIAM*, Philadelphia, PA, 1990.