AN EFFICIENT ALGORITHM FOR FUNCTION APPROXIMATION WITH HINGING HYPERPLANES.

D. Docampo and S. R. Baldomir

Departamento de Tecnoloxías das Comunicacións
E.T.S.E. de Telecomunicación, Universidade de Vigo; 36200-Vigo, Spain
e-mail: ddocampo@tsc.uvigo.es

ABSTRACT

This paper presents computationally efficient algorithms for function approximation with hinging hyperplanes. Approximant units are added one at a time using the method of fitting the residual.

To fit an individual unit we have to solve a sequence of Quadratic Programming problems, an approach which has proven to offer significant advantages over derivative-based search algorithms. Empirical results on synthetic data illustrate the main characteristics of the algorithms.

1. INTRODUCTION

The universal approximation property of feed forward neural networks, established at the end of the past decade, spurred a significant amount of research effort aimed to developing new nonlinear function approximation strategies. Although it is difficult to develop a truly effective method in high dimensions, many interesting results have been obtained using different approaches, including Multilayer Perceptrons, Radial Basis Functions and Wavelet bases. As is shown in [1], the election of a suitable basis function may help in overcoming the *curse of dimensionality* associated with classic approximation procedures. As a result of that effort, we now know that continuous functions on compact subsets of $\mathbb{R}^d$ can be uniformly approximated by linear combinations of several kinds of functions (sigmoidal functions, ridge functions, ramps, hinging hyperplanes, ...). Besides, we also know that this approximation problem can be given a constructive solution where the iterations taking place involve computations in a reduced subset of approximant functions [2].

The problem of these kind of constructive algorithms becomes combinatorial [3]; however, although one must *only* search through a finite number of feasible solutions, the problem cannot be solved efficiently.

In this paper we propose a constructive algorithm, which builds one node at a time, and produces reasonable approximations using simple and efficient strategies. As basic non-linear units we use piecewise linear nodes —hinging hyperplanes (HH)— that have shown very useful in nonlinear function approximation [4].

The rest of the paper is organized as follows: Section 2 describes the kind of constructive algorithms our procedure belongs to, Section 3 reviews the hinging hyperplane approach, Section 4 studies the learning problem associated with HHs, Section 5 shows the performance of our algorithms with some synthetic data, and finally Section 6 closes the paper with a discussion of the results obtained, the conclusions and further work.

2. CONSTRUCTIVE ALGORITHMS.

Constructive algorithms [3] are easy to use, and possess several advantages over alternative methods, namely: computational advantages associated with the constructive approach, relatively easy determination of a suitable network size, and search algorithms that are more robust to parameter selection and initial conditions. Their potential drawbacks are associated with the fact that constructive algorithms generally will not produce networks of minimal size. Furthermore, although the size of the network can be allowed to vary, the function to be approximated may not be efficiently represented by a model with the connectivity scheme imposed by the algorithm.

Jones [2] provided the first constructive solution to the problem of finding finite convex approximations of a given function in a Hilbert space using elements taken from a reduced subset. The proof of his result is itself constructive and thus provides a framework for the development of practical algorithms. In a previous paper [5] we studied the trade-off among the different parameters to guarantee the rate of convergence $O(1/n)$ pre-

The research of D. Docampo was supported by ISTEC, and Xunta de Galicia.

scribed in [2] and [1].

In this paper we propose a very simple constructive solution in which a new approximation of the residual is combined with the previous approximation in such a way that the global error is minimized. The algorithms make use of the aforementioned hinging hyperplanes, updated following new optimization procedures.

3. FUNCTION APPROXIMATION WITH HINGING HYPERPLANES.

Breiman [4] proved that it is possible to use hinging hyperplanes as members of a base in function expansions $f(x) = \sum h_i(x)$ to approximate continuous functions on compact sets, guaranteeing a bounded approximation error

$$\|e_n\| = \|f - \sum_{i=1}^{n} h_i(x)\| \leq (2R)^4 c^2/n$$

where n is the number of nodes in the network, R is the radius of the sphere in which the compact set is contained, and c is such that

$$\int \|w\|^2 |f(w)| dw = c < \infty$$

These results show that it is possible to achieve arbitrarily good approximations of a function by increasing the number of nodes in the network. However, this is not possible in practice for some reason:

1. The information of the function f is provided by way of a sample set $S = \{(x_i, y_i)\}_{i=1}^{N}$ which contains samples of f at a finite number of points. The information in this set is usually insufficient to uniquely characterize the unknown function f, and so the approximation is not optimal.
2. The approximate function must be of finite size.
3. The learning problem can usually not be solved in polynomial time.

Therefore, we can only expect to obtain an estimate of f that approximates f as closely as possible, which is the goal of our method.

3.1. Hinging Hyperplanes

Hinging hyperplanes (HH) [4] are two hyperplanes joined together at a hinge function. The main advantages of using HHs can be summarized as follows:

1. An upper bound on the approximation error is available.
2. HH constitute a piecewise linear model, and linear models have proven to be useful in a large number of problems. Besides, these functions are nonlinear only in one direction, while remaining linear in the others, so they are very useful in the approximation of functions with similar characteristics.
3. They let us use very fast and computationally efficient training algorithms.
4. It is very simple and efficient to create and store these functions.

Some algorithms have been developed to update the HH for a training set. Breiman proposed a Hinge Finding Algorithm (HFA) that implements a gradient descent method after partitioning the data set in two subsets, one for each hyperplane to update. This method shows three important drawbacks: convergence is not guaranteed, the performance of the method depends on the initial conditions, and the final result may not be a global minimum of the error.

Another method to improve the HFA, which uses a damped Newton algorithm to minimize the same error as in the HFA, was developed by Pucar and Sjöberg [6]. Their approach shares some of Breiman's problems; in spite of the improvements, it cannot guarantee the convergence of the method. It also comes with the added disadvantage of requiring a new parameter to be handled by the user, the step of the Newton method. In a subsequent paper [7], they proposed a simultaneous estimation of all parameters in the network, to update the hyperplanes, which suffers from the drawback of requiring an enormous computational burden.

Another drawback of these methods resides in the algorithm used to construct the network: although a new hinge function is updated from the residual of the previous approximation, all base functions must be refitted (*backfitted*) when a new function is added. This means that when a new node is added to the network all previous nodes must be recalculated. This results in a lack of simplicity of the constructive solution.

Just as these algorithms, a great deal of constructive solutions developed nowadays are not constructive in a simple way, because they either use Fourier or random transforms, or rely upon backpropagation procedures which modify previously connected weights.

In this paper we propose a Barron's like constructive solution which can be formulated in three consecutive steps: first compute (1) and then fit (2) the residual; finally update the approximation (3).

$f(\mathbf{x}) \in \mathbb{R}$: Function to be approximated; $\mathbf{x} \in \mathbb{R}^d$.
Initialization: $f_o(\mathbf{x}) = 0$; $e_o(\mathbf{x}) = f(\mathbf{x})$.

```
for  n = 1  to  N  do
(1) $e_n(\mathbf{x}) = f(\mathbf{x}) - f_{n-1}(\mathbf{x})$
(2) $g_n(\mathbf{x}) \approx \arg\min_g ||f(\mathbf{x}) - (1-\lambda_n)f_{n-1}(\mathbf{x}) - \lambda_n g(\mathbf{x})||$
(3) $f_n(\mathbf{x}) = (1-\lambda_n)f_{n-1}(\mathbf{x}) + \lambda_n g_n(\mathbf{x})$
endloop
```

where $e_n(\mathbf{x})$ stands for the residual and λ_n is a parameter which can depend on the norm of the previous residual [1] or can be selected beforehand, as in the elegant modification proposed in [8].

The problem arises in the second step: finding the function g_n that best fits the current residual. It is very difficult to achieve an O(1/n) bound in the approximation error, although Jones [2] proved that it is sufficient to produce a function that is within O(1/n^2) from the optimum.

In the next section we review the hinging hyperplane approach and explain our learning procedure, based on the use of the Sweeping Hinge Algorithm, first introduced by D. Hush and B. Horne [3].

4. LEARNING WITH HINGING HYPERPLANES

By HH we denote a function based on two hinging hyperplanes, defined as follows:

$$h(x,w) = \begin{cases} w_+^T x, & (w_+ - w_-)^T x > 0 \\ w_-^T x, & (w_+ - w_-)^T x \le 0 \end{cases}$$

This function can be viewed as two coupled hyperplanes, whose intersection (hinge) divides the whole space into two different subsets:

$$\begin{aligned} S_+ &= \{x : (w_+ - w_-)^T x > 0\} \\ S_- &= \{x : (w_+ - w_-)^T x \le 0\} \end{aligned}$$

The hinge itself is defined in terms of the parameter w, which stands for the 2D-dimensional vector composed of the two vectors w_+ and w_-. The hinge is the line where the two hyperplanes intersect; points in the hinge verify the following equation: $(w_+ - w_-)^T x = 0$.

In our case the problem is to choose the parameter w which minimizes, in some sense, the error of the approximation of a function f. The information available is a finite set of function samples. If the regression vectors are denoted by x_i, $x_i^T = (x_{i1}, x_{i2}, ..., x_{id})$, then the training set S will consist of the pairs $\{(x_i,\ y_i)\}$, with $y_i = f(x_i)$. The two sets S_+ and S_-, now restricted to the training set S, define the partition of the data space. They are forced to be disjoint by including the points in the hinge in one of the subsets, to make the problem under study well defined. Those points in the hinge will not be excluded, then, from the minimization problem.

We now define the empirical risk of the approximation of the unknown function f in the following way:

$$E_p(w) = \frac{1}{2}\sum_S (y_i - h(x_i,\omega))^2 \qquad (1)$$

To simplify the evaluation of (1), we will make use of the following variables:

$$S_y^2 = \frac{1}{2}\sum_S y_i^2, \quad r_+ = \sum_{S_+} y_i x_i, \quad r_- = \sum_{S_-} y_i x_i,$$
$$R_+ = \sum_{S_+} x_i x_i^T, \quad R_- = \sum_{S_-} x_i x_i^T.$$

After some algebra, we can rewrite expression (1) in a more compact form:

$$\begin{aligned} E_p(w) &= S_y^2 + \frac{1}{2}w_+^T R_+ w_+ + && (2) \\ &+ \frac{1}{2}w_-^T R_- w_- - w_+^T r_+ - w_-^T r_- && (3) \end{aligned}$$

Defining now the matrices:

$$R = \begin{pmatrix} R_+ & 0 \\ 0 & R_- \end{pmatrix} \qquad r = \begin{pmatrix} r_+ \\ r_- \end{pmatrix}$$

we can write (1) as:

$$E_p(w) = S_y^2 + \frac{1}{2}w^T R w - w^T r \qquad (4)$$

4.1. The Minimization Problem

Following [3], we will say that a partition of S into S_+ and S_- is admissible (stable) if the matrix R is definite positive (a true correlation matrix). Minimizing $E_p(w)$ with respect to w is a quadratic problem with a global minima provided that R is definite positive. Nevertheless, when we seek for the least squares solution to $Rw = r$, we actually move the hinge as we vary w; therefore, the initial partition is no longer valid, and expression (3) becomes completely useless. We have to set up the problem in such a way that this minimization process can take place. The solution is to make it iterative; we will have a collection of problems, starting from an initial partition, where each problem can be written as a constrained quadratic optimization:

$$\boxed{\begin{array}{l} \min\limits_w \left\{ \frac{1}{2}w^T R w - w^T r \right\} \\ \text{subject to } Aw < 0 \end{array}}$$

where A is a matrix with as many rows as the number of samples in S, $A = \begin{pmatrix} A^+ \\ A^- \end{pmatrix}$, each row of A^+ and A^- taking the following form

$$a_i^+ = (\,-x_i \quad x_i^T\,)\,, x_i \in S_+$$

$$a_i^- = (x_i \quad -x_i^T) , x_i \in S_-$$

4.2. Moving the hinge

To iterate the algorithm we use the following procedure to move the hinge in search of the optimum value:

1. Initialize the partition and its associated parameters $\{w^1, R_+, R_-, r_+, r_-, A\}$ in such a way that S_+ is composed of all the data points but those for which $(w_+^1)^T x_i$ is minimum.
2. Refine the result by moving the points in the hinge from one region to another until E_p is minimum. Then solve the constrained quadratic optimization problem for the partition encountered.
3. Move the hinge among stable partitions by switching points between S_+, recomputing the set of parameters every time. When all stable partitions have been measured the method keeps the one achieves the minimum E_p.

5. EMPIRICAL RESULTS

We have done extensive simulations to test the approximation capabilities of the method. In this section we show some results obtained using the method with two dimensional signals, without refitting.

The following figures represent the results obtained in the approximation of the two dimensional function $f(x,y) = 1 + \sin(2\pi x)\cos(\pi y)$ using 24 nodes, as well as the evolution of the mean square error with each iteration.

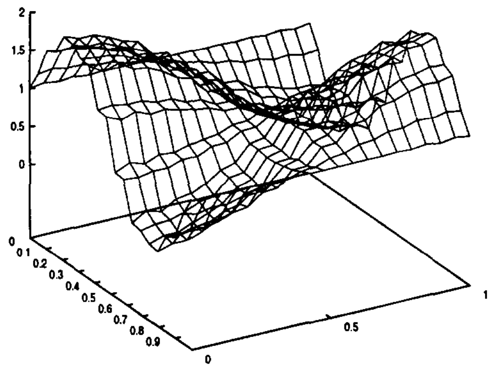

Figure 1: Original function.

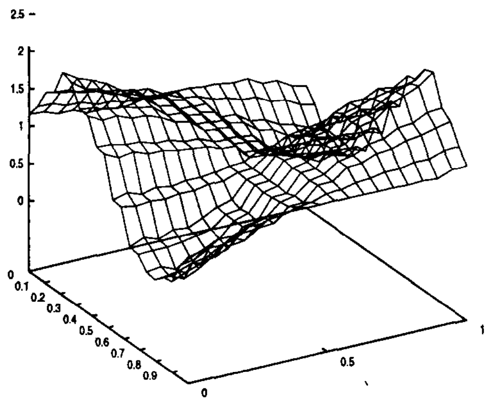

Figure 2: Approximation with 24 nodes.

6. CONCLUSIONS AND FURTHER WORK

From our analysis of the behavior of the method in the simulations made, we come to the following conclusions:

- The minimization technique employed guarantees the convergence of the method.
- It does not depend on the initial conditions, and does not not get trapped in local minima.
- It requires fewer parameters.
- Its computational burden is comparable to Breiman's method.
- Although it is a purely constructive method, it can also benefit from the use of backfitting procedures.
- The error in the approximation decreases as $\frac{1}{n}$.

We have also noticed that the tail of the hyperplanes introduces an error that the method can not compensate with the inclusion of new nodes without introducing a new error in zones that were already adjusted. That is why the greatest part of the error is placed at the peripheral regions of the training set, forcing a large number of nodes to be introduced to refine the approximation in this areas.

Occasionally, the method produces residues symmetrically distributed around zero. When that happens, the approximation does not improve with the introduction of more nodes.

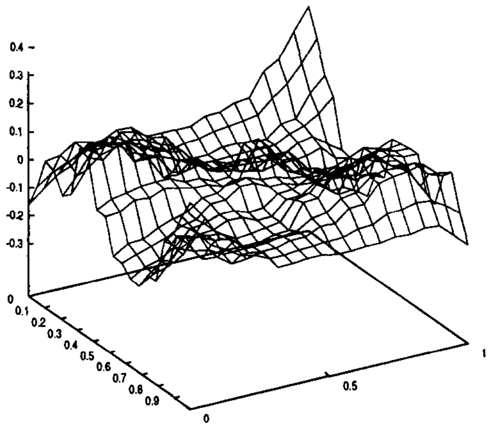

Figure 3: Residual function for 24 nodes.

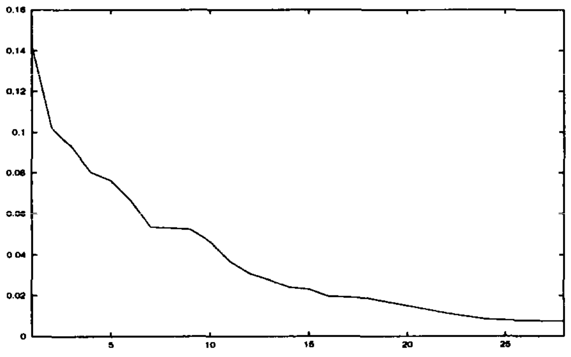

Figure 4: MSSE against number of nodes.

These last drawbacks led us to the conclusion that we should look for a modification of the method to cope for the problems encountered, preserving at the same time the good behavior of the algorithm.

We are now exploring a modification of the method consisting of the truncation of the hinging hyperplanes, in such a way that the new THH function, $th(x,w)$, would be defined as:

$$= \left\{ \begin{array}{ll} w_+, & (w_{l+} - w_{l-})^T x > 0 \text{ and } w_{l+}^T x \geq w_+ \\ w_{l+}x, & (w_{l+} - w_{l-})^T x > 0 \text{ and } w_{l+}^T x < w_+ \\ w_{l-}x, & (w_{l+} - w_{l-})^T x \leq 0 \text{ and } w_{l+}^T x < w_- \\ w_-, & (w_{l+} - w_{l-})^T x \leq 0 \text{ and } w_{l-}^T x \geq w_- \end{array} \right\}$$

where

$$w^T = (\, w_{l+} \quad w_{l-} \quad w_+ \quad w_- \,), \quad w_{l+} \text{ and } w_{l-} \in \mathbb{R}^d.$$

This function is comprised of four hyperplanes joined pairwise continuously at three different hinge locations. These hinges, defined by the components of w induce linear partitions on the input space which is thus divided into four regions.

We have modified the sweeping algorithm to include the search of the three hinges of the function.

Preliminary results confirm our predictions: reduced error in peripheral regions, and no systematic errors around zero. However, the algorithm in its present state requires a large computational burden. Our current work is centered around the intelligent exploration of the optimal partition.

7. ACKNOWLEDGMENTS

The authors want to thank Prof. D.R. Hush for providing both the motivation and the analysis tool to study this problem from a different perspective.

8. REFERENCES

[1] A.R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. Inf. Theory*, 39(3):930–945, 1993.

[2] L.K. Jones. A simple lemma on greedy approximation in hiobert space and convergence rates for projection pursuit regression and neural network training. *The Annals of Statistics*, 20:608–613, 1992.

[3] D. Hush and B. Horne. Efficient algorithms for function approximation with piecewise linear sigmoidal networks. Technical Report EECE96-004, Univ. of New Mexico, 1996.

[4] L. Breiman. Hinging hyperplanes for regression, classification and function approximation. *IEEE Trans. Inf. Theory*, 39(3):999–1013, 1993.

[5] D. Docampo, D.R. Hush, and C.T. Abdallah. Constructive function approximation: theory and practice. In *Intelligent Methods in Signal Processing and Communications*, pages 199–220. Birkhäuser, 1997.

[6] P. Pucar and J. Sjöberg. On the hinge finding algorithm for hinging hyperplanes. Technical report, Linkoping University, Sweden, 1995.

[7] P. Pucar and J. Sjöberg. On the parameterization of hinging hyperplane models. Technical report, Linkoping University, Sweden, 1995.

[8] A. T. Dingankar and I. W. Sandberg. A note on error bounds for approximation in inner product spaces. *Circuits, Systems and Signal Processing*, 15(4):519–522, 1996.

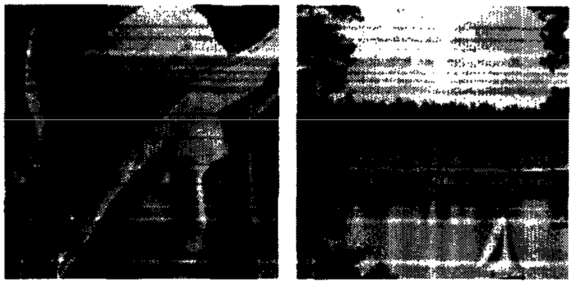

Figure 8: Reconstructed from DCT coded *Left :* Lenna *Right :* Sailboat images.

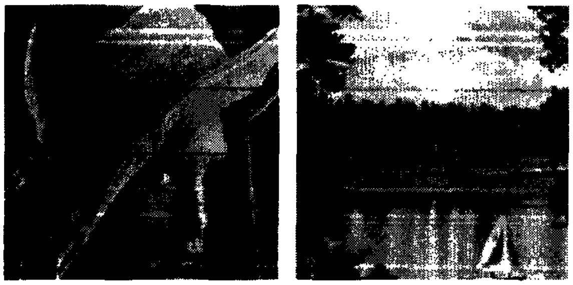

Figure 9: Reconstructed from B-Spline Transform coded *Left :* Lenna image. *Right :* Sailboat images.

Table 2: PSNR and MAE vs bitrate results with quantizer step size = 35.

	DCT		
Image	PSNR (dB)	MAE	bitrate (bpp)
Lenna	32.57	3.50	0.23
Harbour	28.62	6.04	0.48
Bridge	26.85	8.48	0.56
Cablecar	31.54	3.56	0.32
Cornfield	30.50	4.20	0.41
Ball	38.51	0.69	0.11
	B-Spline Transform		
Image	PSNR (dB)	MAE	bitrate (bpp)
Lenna	32.87	3.12	0.19
Harbour	28.82	5.90	0.43
Bridge	26.96	8.35	0.51
Cablecar	31.70	3.53	0.28
Cornfield	30.71	4.20	0.35
Ball	40.22	0.59	0.11

Table 3: PSNR and MAE vs bitrate results with dithering applied in transform domain.

	dithering		
Image	PSNR (dB)	MAE	bitrate (bpp)
Lenna	29.94	6.31	0.19
Harbour	27.56	8.06	0.44
Cablecar	29.16	6.81	0.29
	dithering+smoothing		
Image	PSNR (dB)	MAE	bitrate (bpp)
Lenna	32.57	4.11	0.19
Harbour	28.61	6.61	0.44
Cablecar	31.21	5.05	0.29

Table 4: PSNR and MAE vs bitrate results with quantizer step size = 21, 17,17 for Y, U, and V components, respectively.

	DCT		
Image	PSNR (dB)	MAE	bitrate (bpp)
Lenna	27.13	14.12	0.70
Sailboat	26.44	15.72	1.07
Fruits	29.82	5.64	0.58
House	30.03	10.23	0.49
Peppers	29.63	10.77	0.70
	B-Spline Transform		
Image	PSNR (dB)	MAE	bitrate (bpp)
Lenna	27.13	14.13	0.60
Sailboat	27.66	14.00	0.99
Fruits	31.63	4.95	0.51
House	30.91	9.42	0.41
Peppers	30.71	9.81	0.55