# Constructive Function Approximation: Theory and Practice

**D. Docampo**
**D.R. Hush**
**C.T. Abdallah**

ABSTRACT In this paper we study the theoretical limits of finite constructive convex approximations of a given function in a Hilbert space using elements taken from a reduced subset. We also investigate the trade-off between the global error and the partial error during the iterations of the solution. These results are then specialized to constructive function approximation using sigmoidal neural networks. The emphasis then shifts to the implementation issues associated with the problem of achieving given approximation errors when using a finite number of nodes and a finite data set for training.

## 1 Introduction

It has been shown that continuous functions on compact subsets of $\mathbb{R}^d$ can be uniformly approximated by linear combinations of sigmoidal functions [11, 20]. What was missing from that result is how the error in the approximation is related to the number of sigmoids used. This can be phrased in a more general way as the problem of approximating a given element (function) $f$ in a Hilbert space $H$ by means of an iterative sequence $f_n$, and has an impact in establishing convergence results for projection pursuit algorithms [22], neural network training [5] and classification [12]. Moreover, the fact that one will have to achieve the approximation when samples of $f$ are given has been largely forgotten by most papers which quote the results of [11, 20]. The approximation problem can be given a constructive solution where the iterations taking place involve computations in a reduced subset $G$ of $H$ [22, 5]. This leads to algorithms such as projection pursuit. Convergence of the classical projection pursuit regression techniques [13] however, has been shown to be very slow unless the iterate $f_{n+1}$ is chosen to be an optimal combination of the past iterate $f_n$ and a ridge function of elements of the subset $G$. The bound of the error in this approximation has been refined several times since the initial non-constructive proof given by Maurey, as reported in [23]. Jones [22] 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. His results have been recently refined by Barron [3] and Dingankar [12]. In this paper we report that the rate of convergence obtained in [22] and [5] is the maximum achievable, and, only under some restricted assumptions, the results in Dingankar can be derived as the optimal convex combination to preserve the desired convergence rate.

In the first part of the paper, we formulate the approximation problem in such a way that we can study the limits of the global error, obtain the best possible trade-off between global and partial errors, and give theoretical bounds for the global error when a prespecified partial error is fixed. We then concentrate on the implementation aspects of the problem, specifically, the problem of achieving a certain approximation error using one approximating function at a time. We then discuss some specific sigmoidal functions and algorithms which have been shown to be efficient in solving a particular step of the approximation problem.

The rest of the paper is organized as follows: We start out by reviewing some theoretical results in section 2 where we state the problem and highlight its practical implications. In section 3 we review the theoretical solutions to the problem, and provide the framework under which those solutions can be derived. In section 4 we analyze the limits of the global error and its relation to the partial errors at each step of the iterative process. In section 5 we specialize the constructive functions to sigmoidal functions. Section 6 presents the practical issues associated with implementing a constructive algorithm with an eye towards neural network results. Finally, section 7 presents our conclusions.

## 2 Overview of Constructive Approximation

In this section, we state and present some theoretical results on the constructive approximation problem. In order to state the results in their full generality, let $G$ be a subset of a real or complex Hilbert space $H$, with norm $\|.\|$, such that its elements, $g$, are bounded in norm by some positive constant $b$. Let $\bar{co}(G)$ denote the convex closure of $G$ (i.e. the closure of the convex hull of $G$ in $H$). The first global bound result, attributed to Maurey, concerning the error in approximating an element of $\bar{co}(G)$ using convex combinations of $n$ points in $G$, is the following:

**Lemma 2.1** *Let $f$ be an element of $\bar{co}(G)$ and $c$ a constant such that $c > b^2 - \|f\|^2 = b_f^2$. Then, for each positive integer $n$ there is a point $f_n$ in the convex hull of some $n$ points of $G$ such that:*

$$\|f - f_n\|^2 \leq \frac{c}{n}$$

$\triangle$

The first constructive proof of this lemma was given by Jones [22] and refined by Barron [3]; the proof includes an algorithm to iterate the solution. In the next section, a review of the constructive proof will be presented. We will specifically prove the following in section 3.

**Theorem 2.1** *For each element $f$ in $\bar{co}(G)$, let us define the parameter $\gamma$ as follows:*

$$\gamma = \inf_{v \in H} \sup_{g \in G} \left\{ \|g - v\|^2 - \|f - v\|^2 \right\}$$

*Let now $\delta$ be a constant such that $\delta > \gamma$. Then, we can construct an iterative sequence $f_n$, $f_n$ chosen as a convex combination of the previous iterate $f_{n-1}$ and a $g_n \in G$, $f_n = (1-\lambda) f_{n-1} + \lambda g_n$, such that:*

$$\|f - f_n\|^2 \leq \frac{\delta}{n}$$

**Proof:** See section 3. ■

Note that this new parameter, $\gamma$, is related to Maurey's $b_f^2$, since if we make $v = 0$ in the definition of $\gamma$ we realize that $\gamma \leq b_f^2$.

The relation between this problem and the universal approximation property of sigmoidal networks was clearly established by [22, 5]; specifically, under certain mild restrictions, continuous functions on compact subsets of $\mathbb{R}^d$ belong to the convex hull of the set of sigmoidal functions that one hidden layer neural networks can generate. Moreover, since the proofs are constructive, an algorithm to achieve the theoretical bounds is provided as well.

Other nonlinear approximation techniques have also benefited from the solution to this problem: approximation by hinged hyperplanes [8], projection pursuit regression [28] and radial basis functions [17]. In all these related approximation problems the solution can always be constrained to fall in the closure of the convex hull of a subset of functions (e.g. hinged hyperplanes, ridge functions or radial basis functions in the examples mentioned above).

## 3 Constructive Solutions

For the sake of clarity and completeness, we include here the proof given in [5] and [12].

**Lemma 3.1** *Given $f \in \bar{co}(G)$, for each element of $co(G)$, $h$, and $\lambda \in [0, 1]$:*

$$\inf_{g \in G} \|f - (1-\lambda)h - \lambda g\|^2 \leq (1-\lambda)^2 \|f - h\|^2 + \lambda^2 \gamma \qquad (1.1)$$

**Proof:** The proof of the lemma will be carried out for $f \in co(G)$; it extends to elements in $\bar{co}(G)$ because of the continuity of all the terms involved in the inequalities [10].

Since $f \in co(G)$, there exists a convex combination of elements $g^*$ from $G$, so that $f = \sum_{k=1}^{m} \alpha_k g_k^*$. Let then $g^*$ be a random vector taking values on $H$ with probabilities $P(g^* = g_k^*) = \alpha_k$.

Then: $\quad E(g^*) = f, \quad var(g^*) = E(\|g^* - f\|^2) = E(\|g^*\|^2) - \|f\|^2 \leq b_f^2$.

Additionally, for $v \in H$,

$var(g^*) = var(g^* - v) = E(\|g^* - v - (f - v)\|^2) = E(\|g^* - v\|^2) - \|f - v\|^2$.

Thus, $\forall v \in H$,

$$\begin{aligned} var(g^*) &\leq \sup_{g \in G} \|g - v\|^2 - \|f - v\|^2 \Rightarrow \\ var(g^*) &\leq \inf_{v \in H} \sup_{g \in G} \|g - v\|^2 - \|f - v\|^2 = \gamma. \end{aligned}$$

Now, for $\lambda \in [0, 1]$ and $d \in H$,

$$E(\|\lambda(g^* - f) + d\|^2) = \lambda^2 E(\|g^* - f\|^2) + \|d\|^2 \leq \lambda^2 \gamma + \|d\|^2,$$

and for $\lambda \in [0, 1]$

$$\begin{aligned} \inf_{g \in G} \|f - (1 - \lambda)h - \lambda g\|^2 &\leq E\left(\|(1 - \lambda)h + \lambda g^* - f\|\right)^2 \leq \\ \leq E\left(\|(1 - \lambda)(h - f) + \lambda(g^* - f)\|\right)^2 &\leq (1 - \lambda)^2 \|f - h\|^2 + \lambda^2 \gamma \end{aligned}$$

which concludes the proof of Lemma 3.1. $\triangle$

We can now prove Theorem 2.1, using an inductive argument.

**Proof:** At step 1, find $g_1$ and $\epsilon_1$ so that $\|f - g_1\|^2 \leq \inf_G \|f - g\|^2 + \epsilon_1 \leq \delta$. This is guaranteed by (1.1), for $\lambda = 1$ and $\epsilon_1 = \delta - \gamma$.

Let now $f_n$ be our iterative sequence of elements in $co(G)$, and assume that for $n \geq 2$,

$$\|f - f_{n-1}\|^2 \leq \delta/(n - 1)$$

It is then possible to choose among different values of $\lambda$ and $\epsilon_n$ so that:

$$(1 - \lambda)^2 \|f_{n-1} - f\|^2 + \lambda^2 \gamma \leq \frac{\delta}{n} - \epsilon_n \tag{1.2}$$

At step $n$, select $g_n$ such that:

$$\|f - (1 - \lambda)f_{n-1} - \lambda g_n\|^2 \leq \inf_{g \in G} \|f - (1 - \lambda)f_{n-1} - \lambda g\|^2 + \epsilon_n \tag{1.3}$$

Hence, using (1.1), (1.3) and (1.2), we get: $\|f - f_n\|^2 \leq \dfrac{\delta}{n}$, and that completes the proof of Theorem 2.1. $\blacksquare$

The values of $\lambda$ and $\epsilon_n$ in [5] and [12] are related to the parameter $\alpha$, $\alpha = \delta/\gamma - 1$, in the following way:

$$[5] \quad : \quad \lambda = \frac{\|f - f_{n-1}\|^2}{\gamma + \|f - f_{n-1}\|^2}; \qquad \epsilon_n = \frac{\alpha\delta}{n(n+\alpha)}$$

$$[12] \quad : \quad \lambda = \frac{1}{n}; \qquad \epsilon_n = \frac{\alpha\gamma}{n^2}$$

It is easy to check that, in both cases, $\epsilon_1$ is equal to $\delta - \gamma$ as stated in the proof.

Given that the values of the constant $\lambda$ are different in both cases, we first look for the values of $\lambda$ which make the problem solvable (i.e. feasible values for the constant $\lambda$). Admissible values of $\lambda$ will have to satisfy inequality (1.2) for positive values of $\epsilon_n$; it is easy to show that those values fall in the following interval, centered at Barron's optimal value for $\lambda$:

$$\frac{\|f - f_{n-1}\|^2}{\gamma + \|f - f_{n-1}\|^2} \pm \frac{1}{\gamma + \|f - f_{n-1}\|^2}\sqrt{\|f - f_{n-1}\|^4 - \|f - f_{n-1}\|^2 + \frac{\delta}{n}}$$

To evaluate the possible choices for the bound $\epsilon_n$ we need to make use of the induction hypothesis; introducing it in inequality (1.2), values of $\lambda$ should now satisfy

$$(1-\lambda)^2\frac{\delta}{n-1} + \lambda^2\gamma \leq \frac{\delta}{n} - \epsilon_n$$

In this case, admissible values of $\lambda$ for positive values of $\epsilon_n$ fall in the interval (which always contains the value of $\lambda = 1/n$):

$$\frac{1+\alpha}{n+\alpha} \pm \frac{n-1}{n+\alpha}\sqrt{\frac{\alpha(1+\alpha)}{n(n-1)}}$$

In Figure 1 we show the bounds of this second interval for $\lambda$ as a function of $n$. The bounds are shown in solid lines, the center of the interval using a dotted line, and the value of $\lambda$ in [12] using a dash dotted line. Note how the dash-dotted line approaches the limits of the interval, which results in a poorer value for $\epsilon_n$, as will be shown later.

## *3.1 Discussion*

Since the results presented so far achieve a bound of the global error of $O(1/n)$, and, to construct the solution, a partial error $\epsilon_n$ of $O(1/n^2)$ is the maximum allowed at each step, it is useful to formulate the following questions:

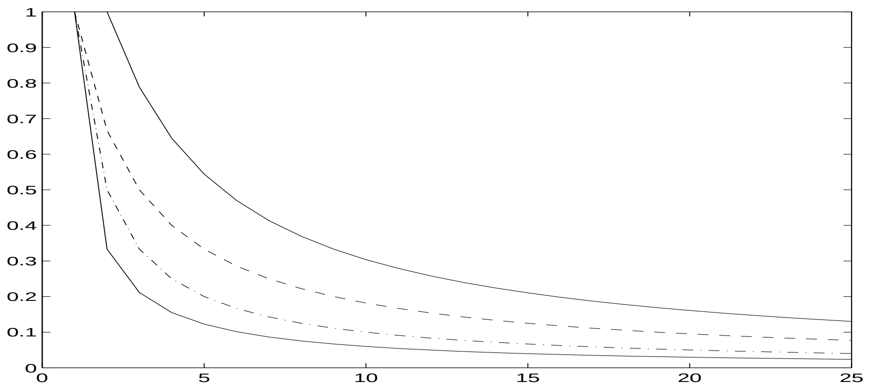

FIGURE 1. admissible values of $\lambda$ for $\alpha = 1$

1. Is there any possibility of achieving a further reduction in the global error using convex combinations of $n$ elements from $G$? What is the minimum bound for the global error assuming $\epsilon_n = 0$ for all $n$?

2. What is the optimal choice of $\lambda$ for a given bound, so that $\epsilon_n$ is maximum, making the quasi-optimization problem at each step easier to solve?

3. For that the optimal choice of $\lambda$ and a prespecified partial error, $\epsilon_n$, what is bound for the global approximation problem?

Based on the assumptions made and in Lemma 3.1, let us formulate the problem again in a more general way: Our objective is to look for a constructive approximation so that the overall error using $n$ elements from $G$ satisfies the following inequality:

$$\|f - f_n\|^2 \leq \frac{\delta}{b(n)} \tag{1.4}$$

$b(n)$ being a function of the parameter $n$ which indicates the order of our approximation (i.e. $b(n) = n$ both in [22] and [12]) and $\delta$ the parameter related to $\gamma$ as defined before.

In what follows we will assume that the iterate $f_n$ will be chosen as a convex combination of the previous iterate $f_{n-1}$ and a point in $G$, $g_n$; this introduces a loss of generality, since other constructive approaches could be devised in order to re-optimize the coefficients of previous elements from $G$ at each step. The facts that $f_n$ is forced to be a convex combination of $n$ elements from $G$, and our algorithm has to be constructive, mean that $f_n$ is in the convex hull of $\{g_1, g_2, \ldots, g_n\}$ and $f_{n-1}$ is in the convex hull of $\{g_1, g_2, \ldots, g_{n-1}\}$, but that does not imply that $f_n$ must be a convex combination of $f_{n-1}$ and $g_n$, as can be easily shown. We leave the more general problem for further investigation and concentrate here on the case where constructiveness of the algorithm is taken as in [22] and [12] to be equivalent to the constraint that, at each step, $f_n$ is in the convex hull of $\{f_{n-1}, g_n\}$.

Before we try to answer the three questions posed at the beginning of this section, let us now set up a framework where the constructive results can be derived.

Let $f_n = (1-\lambda)f_{n-1} + \lambda g_n$, then, in our approximation problem we want to find $\lambda$, $\epsilon_n$, and the function $b(n)$ so that:

$$\begin{aligned}
\|f - f_n\|^2 &\leq \inf_{0<\lambda<1} \inf_{g \in G} \|f - (1-\lambda)f_{n-1} + \lambda g\|^2 + \epsilon_n \\
&\leq \inf_{0<\lambda<1} (1-\lambda)^2 \|f - f_{n-1}\|^2 + \lambda^2\gamma + \epsilon_n \\
&\leq \inf_{0<\lambda<1} (1-\lambda)^2 \frac{\delta}{b(n-1)} + \lambda^2\gamma + \epsilon_n \leq \frac{\delta}{b(n)} \qquad (1.5)
\end{aligned}$$

Since $\delta = (1+\alpha)\gamma$, we can rewrite the last inequality in the following way:

$$\inf_{0<\lambda<1} (1-\lambda)^2 \frac{\delta}{b(n-1)} + \lambda^2\delta + \epsilon_n - \lambda^2\alpha\gamma \leq \frac{\delta}{b(n)}$$

This last expression represents the trade-off between the global error we are trying to achieve $\delta/b(n)$ and the error at each of the subproblems, $\epsilon_n$.

We are going to prove the following: if we set $\epsilon_n = \lambda^2\alpha\gamma$, then, for a given $\lambda$, the best rate of convergence of the approximation which can be achieved, measured in $b(n)$, is the one given in [12] and [5], and the optimal value of $\lambda$ which minimizes $\epsilon_n$ for that best rate of convergence is precisely the value given in [12]. To see that, let's introduce the value of $\epsilon_n$ in (1.5), then:

$$(1-\lambda)^2 \frac{\delta}{b(n-1)} + \lambda^2\delta \leq \frac{\delta}{b(n)}$$

Hence,

$$(1-\lambda)^2 + b(n-1)\lambda^2 \leq \frac{b(n-1)}{b(n)}$$

and then:

$$P(\lambda) = \lambda^2((1 + b(n-1)) - 2\lambda + 1 - \frac{b(n-1)}{b(n)} \leq 0 \qquad (1.6)$$

$P(\lambda)$ has to have a discriminant greater than or equal than 0 for the inequality (1.6) to hold. So,

$$1 - (1 + b(n-1)) \left(1 - \frac{b(n-1)}{b(n)}\right) \geq 0$$

and then, finally:

$$\begin{aligned} b(n) &\geq (1+b(n-1))\,(b(n)-b(n-1)) \Leftrightarrow \\ b(n-1) &\geq b(n-1)\,(b(n)-b(n-1)) \Leftrightarrow \\ b(n) &\leq 1+b(n-1) \end{aligned} \tag{1.7}$$

Inequality (1.7) proves that, under the assumption that $\epsilon_n = \lambda^2\alpha\gamma$, there is no better rate of convergence using these kind of convex constructive solutions that the one obtained in references [22] and [12], since the maximum rate is obtained when

$$b(n) = 1 + b(n-1) \Rightarrow b(n) = b(1) + n - 1 = n \tag{1.8}$$

Furthermore, for this rate of convergence there is only one zero of the function $P(\lambda)$, namely, $\lambda = (1/n)$ which is the optimal value and coincides with the one provided in Dingankar's algorithm.

We will next answer the questions posed at the beginning of this section, concerning the limits and bounds of the approximation.

## 4 Limits and Bounds of the Approximation

If we look back at expression (1.5), we will notice that, after using Lemma 3.1, we have at each step a quadratic problem in $\lambda$, which consists of minimizing

$$Q(\lambda_n) = (1-\lambda_n)^2 \frac{\delta}{b(n-1)} + \lambda_n^2\gamma$$

provided that the induction hypothesis (1.4) is satisfied for $k < n$. We have introduced the notation $\lambda_n$ to stress the variation of this parameter along the iterative process.

Taking derivatives, we get

$$\lambda_n\gamma = (1-\lambda_n)\frac{\delta}{b(n-1)} \Rightarrow$$

$$\lambda_n = \frac{(1-\lambda_n)\delta}{\gamma b(n-1)} = \frac{1+\alpha}{1+\alpha+b(n-1)} \tag{1.9}$$

Hence, we get the following expression of the optimal error bound:

$$\begin{aligned} \|f-f_n\|^2 &\leq (1-\lambda_n)^2\left[\frac{\delta}{b(n-1)} + \frac{(1+\alpha)\delta}{b^2(n-1)}\right] + \epsilon_n \\ &= \delta\left(\frac{b^2(n-1)}{1+\alpha+b(n-1)}\right)\left(\frac{1+\alpha+b(n-1)}{b^2(n-1)}\right) + \epsilon_n \\ &= \frac{\delta}{1+\alpha+b(n-1)} + \epsilon_n = \frac{\delta}{b(n)} \end{aligned} \tag{1.10}$$

From (1.10) we can write the following expression for $b(n)$ and $\epsilon_n$:

$$\frac{1}{b(n)} = \frac{1}{1+\alpha+b(n-1)} + \frac{\epsilon_n}{\delta} \tag{1.11}$$

and then

$$\epsilon_n = \frac{\delta}{b(n)(1+\alpha+b(n-1))}\left[1+\alpha+b(n-1)-b(n)\right] \tag{1.12}$$

From this last expression we conclude that there is a fundamental limitation in the rate of convergence that can be achieved under the hypothesis made so far, namely:

$$b(n)-b(n-1) \leq 1+\alpha = \frac{\delta}{\gamma}$$

## *4.1 Minimum Global Error*

Assuming that we can solve the partial approximation problems at each step of the iteration, so $\epsilon_n = 0, n \geq 1$, then

$$b(n) = 1+\alpha+b(n-1) \quad \Rightarrow \quad b(n) = n(1+\alpha)$$

provided that we make $b(1) = 1+\alpha$, which means that we should find an element $g_1$ in $G$ so that

$$\|f-f_1\|^2 \leq \frac{\delta}{1+\alpha}$$

which is guaranteed by Lemma 3.1. Hence, the best rate of convergence that can be obtained follows the law $c/n$, since

$$\frac{\delta}{n(1+\alpha)} = \frac{\gamma}{n}$$

We have then reached the minimum value of the constant $c$, namely $c = \gamma$.

Note that for this minimum to be reached we have

$$\lambda_n = \frac{1+\alpha}{(1+\alpha)n} = \frac{1}{n}$$

so the optimal convex combination would be the average of $n$ elements from $G$, as in [12].

We have then answered the first of our questions. We now examine the trade-off between the global error and $\epsilon_n$. Specifically, we will find the minimum global error for a specified partial error, $\epsilon_n$, and the maximum bound we can place on the partial error for a prespecified rate of convergence.

### *4.2 Fixing $\epsilon_n$*

Given the nonlinear character of the recursion involved in (1.11), there is no analytical procedure to find a closed expression for $b(n)$. However, we can compute the bound of the approximation following the flow diagram of the optimal procedure, and from it derive some asymptotically results.

1. Select a constant $\delta$ such that $\delta > \gamma$; let $\delta = (1+\alpha)\gamma$.
2. Find $g_1 \in G$ so that $\|f - g_1\|^2 \le \delta$. Set $f_1 = g_1$.
3. For $n > 1$, evaluate:
   (a) $\lambda_n = (1+\alpha)/\left(1+\alpha+b(n-1)\right)$ from (1.9)
   (b) Find $g_n \in G$ so that
   $$\|f-(1-\lambda_n)f_{n-1}-\lambda_n g_n\|^2 \le \inf_G \|f-(1-\lambda_n)f_{n-1}-\lambda_n g\|^2 + \epsilon_n$$
   (c) Make $f_n = (1-\lambda_n)f_{n-1} + \lambda_n g_n$
   (d) Compute $b(n)$ from (1.11)

In order to make the appropriate comparisons with previous results, we will set $\epsilon_n = (\alpha\gamma/n^2)$, as in [12]. Then, again under the induction hypothesis,

$$\frac{1}{b(n)} = \frac{1}{1+\alpha+b(n-1)} + \frac{\alpha}{(1+\alpha)n^2}$$

To predict the asymptotic behavior of $b(n)$, let us assume that, at step $n-1$, $b(n-1) \ge \beta(1+\alpha)(n-1)$, we will prove then that, for some values of the constant $\beta$, we can imply that also $b(n) \ge \beta(1+\alpha)n$.

Since $b(n-1) \ge \beta(1+\alpha)(n-1)$, we have:

$$\begin{aligned}
\frac{1}{b(n)} &\le \frac{1}{(1+\alpha)(1+\beta(n-1))} + \frac{\alpha}{(1+\alpha)n^2} \Rightarrow \\
\frac{1+\alpha}{b(n)} &\le \frac{1}{1+\beta(n-1)} + \frac{\alpha}{n^2} \Rightarrow \\
\frac{b(n)}{n(1+\alpha)} &\ge \frac{n(1+\beta(n-1))}{n^2+\alpha(1+\beta(n-1))} \Rightarrow \\
\frac{b(n)}{n(1+\alpha)} \ge \beta \;&\Leftrightarrow\; n(1-\beta) \ge \beta\alpha(1+\beta(n-1)) \Leftrightarrow \\
n(1-\beta-\beta^2\alpha) &\ge \beta\alpha(1-\beta)
\end{aligned}$$

This last inequality is asymptotically fulfilled for any value of $\beta$ such that:

$$0 \le \beta \le \frac{\sqrt{4\alpha+1}-1}{2\alpha}$$

Then, for the value of $\epsilon_n$ selected in [12], the asymptotic value for $b(n)$ is:

$$b(n) = (1+\alpha)n\frac{\sqrt{4\alpha+1}-1}{2\alpha}$$

which is a better rate than the one obtained in [12].

In Figure 2 we show $b(n)$ as a solid line, and the straight lines $l(n) = n$ corresponding with the rate in [12], dotted line, and the predicted asymptotic behavior of $b(n)$. The figure clearly supports the asymptotic results, and shows that the constant $\lambda_n$ found (1.9) always results in a better convergence rate than [12]. The gap between the two lines would be bigger for larger values of the constant $\alpha$; in other words, the larger the constant $\delta$ the worse the convergence rate achieved using $\lambda = 1/n$.

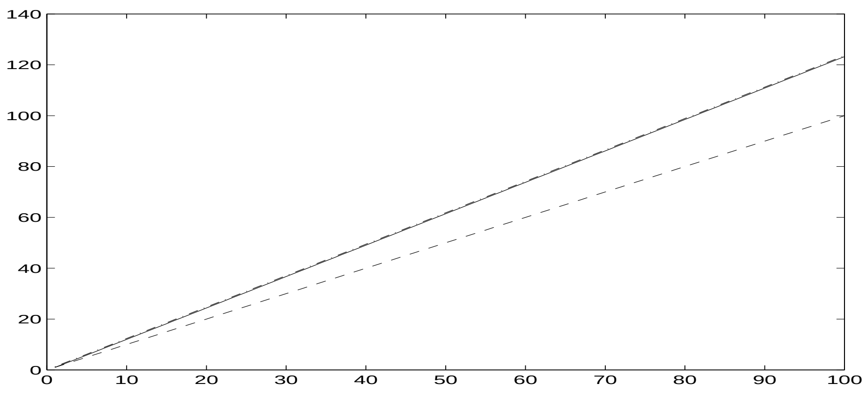

FIGURE 2. Optimal convergence rate for $\alpha = 1$

## *4.3 Fixing the rate of convergence*

The remaining problem, namely: given the optimal value of $\lambda$ find the maximum $\epsilon_n$ for a fixed convergence rate, thus making the quasi-optimization problem at each step easier to solve, was already explicitly solved in (1.12). Again, to show how our results compare with [5] and [12], we will assume that our desired rate of convergence is given by $b(n) = n$.

The value $\lambda_n = (1+\alpha)/(n+\alpha)$ solves the optimization problem, and:

$$\epsilon_n = \frac{\alpha\delta}{n(n+\alpha)} \tag{1.13}$$

This is the best upper bound we can achieve for the partial error at each step of the iteration process. It is easy to show that it coincides with Barron's bound, and is always greater than the bound found in [12].

Now, in Figure 3 we show the bound $\epsilon_n$ for $n = 5$ and $\gamma = 1$, as a function of $\alpha$. The optimal bound is shown using the solid line, while the bound from [12] is shown using a dotted line.

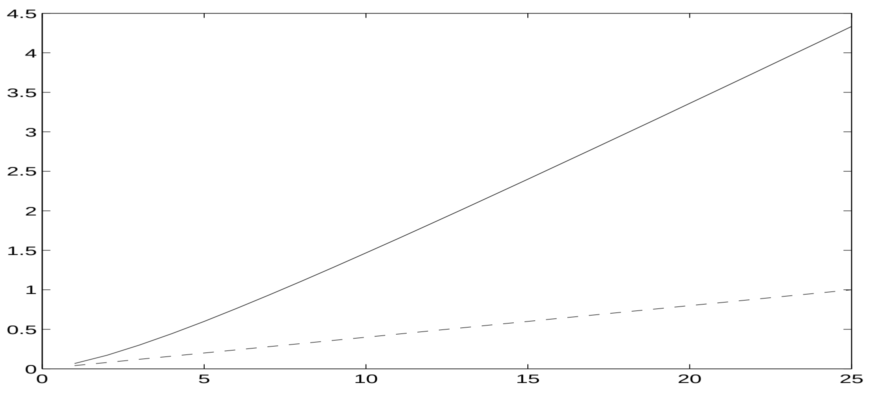

FIGURE 3. $\epsilon_n$ for $n = 5$

## 5 The Sigmoidal Class of Approximators

When discussing neural networks, we are typically referring to a system built by linearly combining a large collection of simple computing devices (i.e., nodes), each of which performs a nonlinear transformation $\sigma$ (in general a sigmoid function) on its inputs [18]. A sigmoid is defined here as a bounded function $\sigma(x)$. It is now known that a 1-hidden layer static network, whose nodes are sigmoidal is capable of approximating an arbitrary (continuous) function. Many proofs of this result have appeared of which we recall the ones in [11, 20]. Until recently, these proofs have used the Stone-Weirestrass theorem and required the continuity or even differentiability of the sigmoid (or nonlinearities) in the neural net. Chen et al. [9], building on the research of Sandberg [25, 26, 27] have recently shown however that all is needed is the boundedness of the sigmoid building block. Table 5 is taken from [9] and summarizes some available results for the approximation of functions. The set $K$ denotes a compact subset of $\mathbb{R}^n$. Note that even those results labeled "constructive" still ignore the facts associated with the training algorithm and the available data.

The set of popular sigmoids include the *hardlimiting threshold* or *Heaviside* function shown in Figure 4(a):

$$\sigma_H(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{1.14}$$

| Reference | Activation Function | Approximation In | Proof |
|---|---|---|---|
| [11] | Continuous Sigmoid | $C[K]$ | Existential |
| [11] | Bounded Sigmoid | $L_p[K]$ | Existential |
| [20] | Monotone Sigmoid | $C[K]$ | Constructive |
| [9] | Bounded Sigmoid | $C[\mathbb{R}^n]$ | Constructive |

TABLE 1.1. Approximation Results

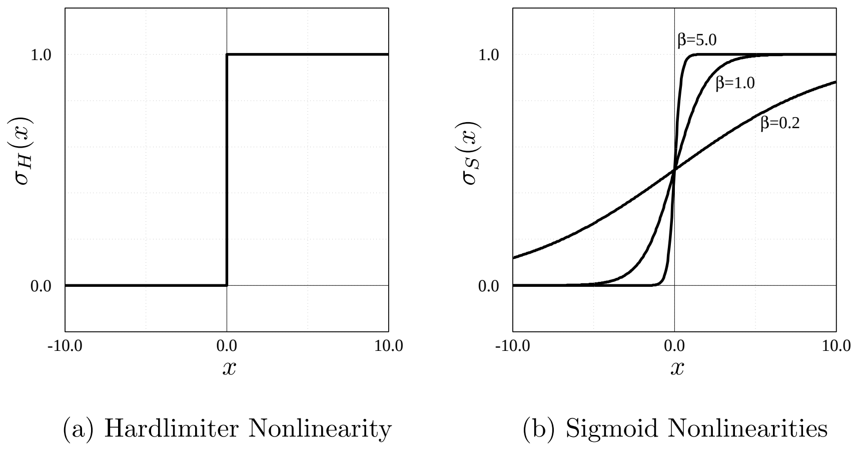

FIGURE 4. Typical Nonlinearities.

In order to derive certain learning techniques, a continuous nonlinear activation function is often required. For example, gradient descent techniques typically require that the sigmoid be differentiable [2]. Thus the threshold function is commonly approximated using the *sigmoid* function shown in Figure 4(b):

$$\sigma_S(x) = \frac{1}{1+e^{-\beta x}} \tag{1.15}$$

The gain of the sigmoid, $\beta$, determines the steepness of the transition region. Note that as the gain approaches $\infty$, the sigmoid approaches a hardlimiting threshold. Often the gain is set equal to one, and $\beta$ is omitted from the definition in equation (1.15). Later in this paper, we shall use the ramp equation which is another sigmoid defined as

$$\sigma_r(\mathbf{x}) = \left\{ \begin{array}{rl} 0, & z \leq \alpha \\ (z-\alpha)/(\beta-\alpha), & \alpha \leq z \leq \beta \\ 1, & \beta \leq z \end{array} \right. \tag{1.16}$$

```
Initialization:
$f_0(\mathbf{x}) = 0$
for  $n = 1$  to  $n_{max}$  do
    1. Compute Residual:    $e_n(\mathbf{x}) = f(\mathbf{x}) - f_{n-1}(\mathbf{x})$
    2. Fit Residual:        $g_n(\mathbf{x}) = \arg\min_{g \in G} \|e_n(\mathbf{x}) - g(\mathbf{x})\|$
    3. Update Estimate:     $f_n(\mathbf{x}) = \alpha f_{n-1}(\mathbf{x}) + \beta g_n(\mathbf{x})$
        where $\alpha$ and $\beta$ are chosen to minimize $\|f(\mathbf{x}) - f_n(\mathbf{x})\|$
endloop
```

FIGURE 5. Iterative Approximation Algorithm (IIA).

## 6 Practical Considerations

In section 2 we recalled that it is possible to approximate an unknown function $f \in \bar{co}(G)$ with a convex combination of functions $g_i \in G$, $i = 1, 2, ..., n$ such that the approximation error is bounded by $\delta/n$. More importantly it was shown that the functions $g_i$ can be found *constructively*, i.e. one at a time. The proof of this result was itself constructive, and thus provides a framework for the development of an algorithm for building these approximations. The purpose of this section is to explore such algorithms. In doing so, our primary concerns are with the *feasibility* and *computational complexity* of these algorithms. Specific statements regarding these two issues are often not possible without considering a specific class of approximating functions. For this purpose we have chosen the class of sigmoidal functions described in the previous section.

Perhaps the most straightforward algorithm that can be derived from the proof in section 3 is the *iterative approximation algorithm* (IIA) shown in Figure 5 [5]. This algorithm is attractive in that the main loop contains only three steps, each of which is conceptually quite simple. However, implementation of this algorithm is not so straightforward. The second step in particular can be quite difficult. This step requires that we find the function $g_n$ that best fits the current residual. Even though this step can be difficult, it is certainly more manageable than finding all $n$ of the $g_i$ simultaneously, which is the more popular (non-constructive) approach used in neural networks.

The final model produced by the IIA algorithm can be expressed in the form

$$f_n(\mathbf{x}) = \sum_{i=1}^{n} a_i g_i(\mathbf{x}) \tag{1.17}$$

where the $a_i$ coefficients are simple deterministic functions of the $\alpha$ and $\beta$ values computed at each step of the IIA. Thus, the model produced by this algorithm has the familiar "linear combination of basis functions" form, where $\{g_i\}_{i=1}^n$ forms the basis set. In this context the $g_i$ are referred to as *tunable* basis functions because they are "tuned" to the data in Step 2 of the algorithm. This is in contrast to the more conventional basis function approach (e.g. polynomial expansions) where the basis are *fixed* a priori, and only their relative weightings (i.e. the $a_i$'s) are tunable.

Let us look more closely at the three steps of the IIA. The first step involves a simple subtraction which is trivial. The third step also turns out to be quite simple when the $L_2$ norm is used. In this case the solution for $\alpha$ and $\beta$ has a simple closed form. The second step however, generally does *not* have a closed form solution. This step involves a search over the function space $G$ for the best fitting member $g_n$. The intrinsic difficulty of this search depends largely on the manner in which the members of $G$ are parameterized, but because they are *nonlinear* and *tunable* there is generally no closed form solution, so the search must be performed algorithmically. Unfortunately, even algorithmic solutions are not guaranteed to produce the optimal function in an efficient manner. All is not lost however, since it is often possible to produce "good approximations" (i.e. near optimal functions) in a computationally efficient manner. This may be adequate, since the function produced in Step 2 does not need to be optimal to achieve the $O(1/n)$ bound in approximation error. In section 3 we saw that it was sufficient for Step 2 to produce a function that is within $O(1/n^2)$ of the optimum. A question remains however, as to whether or not this can actually be achieved. Although efficient algorithms exist for Step 2, we know of no *proof* that any such algorithm can produce functions that satisfy the $O(1/n^2)$ tolerance in polynomial time. To our knowledge this is an open question.

In practice it is common to use a *refitting* (or *backfitting*) procedure to "fine tune" the result of the IIA. This procedure can compensate somewhat for the suboptimal result produced at Step 2, and also to some degree for the possible limitations due to the constructive nature of the IIA. A typical refitting procedure is shown in Figure 6. The basic idea is to refit each basis function, one at a time, to the residual formed from the approximation using the other $n-1$ basis functions. This algorithm has the same attributes as the IIA: it optimizes individual basis functions by fitting them to a residual, and then reintegrates them into the overall fit. It differs from the IIA in that the residual is computed differently, and that the starting point for each refitting is usually close to its final point. This means that the search in Step 3 is generally very fast compared to its counterpart in Step 2 of the IIA. Because of this, refitting usually runs much faster than the IIA.

## *6.1 Projection Pursuit Methods*

This section presents a brief introduction to *projection pursuit* (PP) methods [21]. Projection pursuit can be thought of as a general methodology

```
for $i = 1$ to $n$ do
    1. Compute $f_{n-1}$:        $f_{n-1}(\mathbf{x}) = \sum_{j \neq i}^{n} a_j g_j(\mathbf{x})$
    2. Compute Residual:         $e_{n-1}(\mathbf{x}) = f(\mathbf{x}) - f_{n-1}(\mathbf{x})$
    3. Fit Residual:             $g_i(\mathbf{x}) = \arg\min_{g \in G} \|e_{n-1}(\mathbf{x}) - g(\mathbf{x})\|$
    4. Update Estimate:          $f_n(\mathbf{x}) = \sum_{j=1}^{n} a_j g_j(\mathbf{x})$
       where $\{a_i\}$ are chosen to minimize $\|f(\mathbf{x}) - f_n(\mathbf{x})\|$
endloop
```

FIGURE 6. Refitting Algorithm.

for data exploitation that can be used in a variety of problem domains including regression, classification, density estimation and data exploration [16, 14, 15, 21]. When used for regression, PP encompasses a popular class of algorithms used to solve Step 2 of the IIA. The motivation for projection pursuit regression (PPR) is the following. Nonlinear regression can be performed accurately and robustly in lower dimensions (e.g. 1 or 2) using a wide variety of techniques (e.g. polynomials, splines, Parzen windows, etc.). However, the natural extension of these techniques to higher dimensional problems is hampered by the *curse of dimensionality*. This curse manifests itself in a variety of ways, one of which is in the rapid growth in the number of free parameters associated with the model. For example, in dimension $d$ the number of free parameters associated with a polynomial model of degree $q$ is $O(d^q)$. This number can become quite large for even modest values of $q$ and $d$. Projection Pursuit methods attempt to circumvent the curse of dimensionality by projecting the input to a 1–dimensional space before fitting a nonlinear regression function. The projection is usually linear (or affine) and thus defines a "direction of pursuit" in the original $d$–dimensional space. The key to success with PPR methods is in finding good projection directions. One version of the PPR algorithm is shown in Figure 7 [21]. The univariate fit in Step 2 can be performed using any number of nonlinear regression methods (cubic splines are a popular choice). Finding the new projection in Step 3 is the most difficult step in the algorithm. This is a nonlinear optimization problem that can be approached in a variety of ways. We will explore this step in more detail through a specific example in the next section, but for now suffice it to say that it can be computationally expensive. Although PP has become a very popular technique for nonlinear regression, the complexity of Step 3, as well as the potentially large number of iterations required for convergence of the main loop, make it a computationally expensive procedure. In the next section we show how PPR can be used with neural networks. Initially this amounts to little more than using the logistic function for $\phi$ in PPR. But as we shall

```
Initialization:
Choose an initial projection, $\mathbf{w}_0$
repeat
    1. Project Data:                                   $u = \mathbf{w}_k^T \mathbf{x}$
    2. Univariate Fit (find $\phi$ with $\mathbf{w}$ fixed):   $\phi_k = \arg\min_{\phi} \|y - \phi(u)\|$
    3. Update Projection (find $\mathbf{w}$ with $\phi$ fixed):   $\mathbf{w}_{k+1} = \arg\min_{\mathbf{w}} \|y - \phi_k(\mathbf{w}^T \mathbf{x})\|$
until ($\mathbf{w}_k$ converges) ;
```

FIGURE 7. The Projection Pursuit Regression Algorithm.

see, the computational efficiency can be improved dramatically if we use piecewise continuous functions (called "ramp functions") instead.

## *6.2 Projection Pursuit with Neural Networks*

We use the following notation to simplify our development in this section. The symbol $\tilde{\mathbf{x}}$ will be used to represent input vectors $\mathbf{x}$ that have been augmented with a 1 in the first position, i.e.

$$\tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix} \tag{1.18}$$

Similarly, $\tilde{\mathbf{w}}$ will be used to represent weight vectors $\mathbf{w}$ that have been augmented with a "bias" weight in the first position

$$\tilde{\mathbf{w}} = \begin{bmatrix} w(0) \\ \mathbf{w} \end{bmatrix} \tag{1.19}$$

The dimension of these augmented vectors is $d + 1$ and is denoted $\tilde{d}$.

In neural networks the most popular tunable basis function is arguably the logistic function,

$$\sigma_l(\mathbf{x}) = (1 + e^{-(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}})})^{-1} \tag{1.20}$$

This function is smooth, bounded and parameterized by the weight vector $\tilde{\mathbf{w}}$. If we wish to use the logistic function in the PPR algorithm it needs to be scaled and shifted so that it can better fit functions with arbitrary range and position. These scaled and shifted logistic functions form the members of $G_s$,

$$g(\mathbf{x}) = a_0 + a_1 \sigma_l(\mathbf{x}) = a_0 + a_1 (1 + e^{-(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}})})^{-1} \tag{1.21}$$

Although one could argue that this scaling and shifting is not needed, since it can be accounted for by the linear weights in (1.17), it is more convenient from the standpoint of algorithmic development to include them separately as we have done here. When viewed from the projection pursuit perspective, $\tilde{\mathbf{w}}$ plays the role of the projection vector and $g$ plays the role of the regression function $\phi$. In this case, Step 2 of the PPR algorithm involves updating the coefficients $a_0$ and $a_1$.

Using $g$ in (1.21) as the basis, the optimization problem in Step 2 of the IIA (or equivalently Steps 2 and 3 of PPR) takes on the form

$$\theta_l^* = \{a_0, a_1, \tilde{\mathbf{w}}\}^* = \arg\min_{\theta_l} \int \left(e_n(\mathbf{x}) - a_0 - a_1\sigma_l(\mathbf{x})\right)^2 d\mu(\mathbf{x}) \qquad (1.22)$$

where $\mu(\mathbf{x})$ is a suitable probability measure on the input space. In practice we don't have access to $e_n(\mathbf{x})$ (recall that $e_1(\mathbf{x}) = f(\mathbf{x})$), only samples of this function at a finite number points in the input space. This forces us to consider a somewhat different optimization problem, where we seek the $\theta_l$ that minimizes the error over the sample data set $\{\mathbf{x}_i, e_n(\mathbf{x}_i)\}, i = 1, 2, ...N$. This new optimization problem takes on the form

$$\theta_{l,N}^* = \{a_0, a_1, \tilde{\mathbf{w}}\}_N^* = \arg\min_{\theta_l} \sum_{i=1}^{N} \left(e_n(\mathbf{x}_i) - a_0 - a_1\sigma_l(\mathbf{x}_i)\right)^2 \qquad (1.23)$$

The solution to this optimization problem is generally different from the solution to (1.22). This in turn introduces error into our estimate of $g_n$. If we let $g_{n,N}$ represent the function parameterized by $\theta_{l,N}^*$ in (1.23), then there is an error of the form

$$e_{n,N} = g_{n,N} - g_n \qquad (1.24)$$

due to the fact that $g$ is estimated using a finite number of samples. This error is referred to as the *estimation error* [4]. The extent to which this error becomes significant depends largely on the quantity and richness of the sample data. For a single sigmoidal function Barron has shown that the estimation error is bounded by the following expression [4]

$$\|e_{n,N}\| = O(d \log N/N) \qquad (1.25)$$

where all $N$ samples are assumed to be independent and identically distributed (IID). This result assumes "perfect learning", i.e. that $\theta_{l,N}^*$ is the global optimum of (1.23). Thus, if we hope to estimate $g_n$ closely enough so that $e_{n,N}$ is within $O(1/n^2)$ of $e_n$, Barron's result tells us that the number of samples $N$ should satisfy $N/\log N = \Omega(dn^2)$. With this in mind we turn to the task of actually solving for $\theta_{l,N}^*$ in (1.23).

The solution to (1.23) has no closed form. This is easy to see because $\sigma_l$ is a nonlinear function of the weight vector $\tilde{\mathbf{w}}$. But (1.23) is an unconstrained nonlinear optimization problem with a differentiable criterion, and thus lends itself to a wide variety of *local descent* search methods. The most popular method (the backpropagation algorithm [24]) uses a stochastic gradient approach. A number of more sophisticated algorithms such as Levenberg–Marquart, conjugate gradient and modified–Newton have also been proposed (see [6] for an overview of these methods). Unfortunately, the characteristics of the criterion in (1.23) can make it very difficult to optimize. Local minima can be a real problem. Their exact number and location depend largely on the data set, but it is possible to have up to an exponential number of them [1]. In addition, convergence is generally quite slow for local descent methods, and a robust stopping criteria for these searches is especially difficult to come by (see [19] for a perspective on why this is true). Although it is difficult to avoid all of these problems entirely, their degree of severity can be reduced tremendously by making a slight change in the basis function.

By replacing the logistic function with the *ramp function* (described in (1.16) and repeated below), Breiman and Friedman were able to develop a search algorithm that runs orders of magnitude faster than algorithms like backpropagation [7]. A ramp function is defined as follows. Let $z = \mathbf{w}^T\mathbf{x}$, then

$$\sigma_r(\mathbf{x}) = \begin{cases} 0, & z \le \alpha \\ (z-\alpha)/(\beta-\alpha), & \alpha \le z \le \beta \\ 1, & \beta \le z \end{cases} \tag{1.26}$$

This function can be viewed as a piecewise continuous approximation to the logistic function defined in (1.20). It is a member of $G_s$ defined in section 5 and thus satisfies all the properties necessary for the approximation (and estimation) error bounds presented previously.

Following the previous development for the logistic function, the members of $G_s$ derived from the ramp function are of the form

$$g(\mathbf{x}) = a_0 + a_1\sigma_r(\mathbf{x}) \tag{1.27}$$

The optimization problem (corresponding to Step 2 of the IIA) then takes on the form

$$\theta^*_{r,N} = \{a_0, a_1, \alpha, \beta, \mathbf{w}\}^*_N = \arg\min_{\theta_r} \sum_{i=1}^{N} \left[e_n(\mathbf{x}_i) - a_0 - a_1\sigma_r(\mathbf{x}_i)\right]^2 \tag{1.28}$$

subject to the constraint

$$\|\mathbf{w}\| = 1 \tag{1.29}$$

An algorithm for approximating $\theta^*_{r,N}$ is shown in Figure 8 [7]. Conceptually this algorithm partitions the input samples into three sets, $S_-$ , $S_l$ and $S_+$, and performs a least-squares fit to the samples in each set separately. The fit to $S_-$ and $S_+$ is a constant ($0^{th}$ order fit), while the fit to $S_l$ is a hyperplane ($1^{st}$ order fit). It then uses the result of these fits to re–partition the data into new sets $S_-$ , $S_l$ and $S_+$. This process is continued until the partitions converge. The bulk of the work performed in the main loop of this

```
{ Let $y_i = e_n(\mathbf{x}_i)$ below. }
Initialization:
Make initial guesses for $\mathbf{w}$, $\alpha$ and $\beta$
repeat
    1. Compute $z_i = \mathbf{w}^T\mathbf{x}_i$, $i = 1, 2, ..., N$.
    2. Partition the input data into $S_-$, $S_l$ and $S_+$:
            $S_- = \{(\mathbf{x}_i, y_i) : z_i < \alpha\}$
            $S_l = \{(\mathbf{x}_i, y_i) : \alpha \le z_i \le \beta\}$
            $S_+ = \{(\mathbf{x}_i, y_i) : \beta < z_i\}$
    3. Perform Least-Squares Fit to $S_-$, $S_l$ and $S_+$:
            $\mathbf{R} = (\sum_{S_l} \tilde{\mathbf{x}}_i\tilde{\mathbf{x}}_i^T)/N_l$
            $\mathbf{r} = (\sum_{S_l} \tilde{\mathbf{x}}_i y_i)/N_l$
            $\beta_- = (\sum_{S_-} y_i)/N_-$
            $\beta_+ = (\sum_{S_+} y_i)/N_+$
            $\tilde{\mathbf{w}} = \mathbf{R}^{-1}\mathbf{r}$
    4. Update $\alpha$ and $\beta$:
            $\alpha = (\beta_- - \tilde{\mathbf{w}}(0))/\|\mathbf{w}\|$
            $\beta = (\beta_+ - \tilde{\mathbf{w}}(0))/\|\mathbf{w}\|$
    5. Normalize $\mathbf{w}$:
            $\mathbf{w} = \mathbf{w}/\|\mathbf{w}\|$
until ($\mathbf{w}$,$\alpha$,$\beta$ converge) ;
Compute the bias and scale parameters:
$a_0 = \beta_-$
$a_1 = \beta_+ - \beta_-$
End
```

FIGURE 8. Breiman/Friedman Ramp Function Algorithm.

algorithm involves least-squares fitting, a process for which very efficient and numerically robust algorithms exist. Experience with this algorithm shows that it converges very quickly and scales well to higher dimensions. Its efficiency and robustness is often far superior to that of local descent algorithms (like backpropagation) used with the logistic basis.

## 7 Conclusions

In this paper, we have reviewed some theoretical results on constructive function approximation. We have specifically set up a framework where constructive algorithms based on convex combinations of elements taken from a subset of a Hilbert space can be analized. We have obtained the optimal values for the coefficients in the convex expansions to guarantee a desired convergence rate. We have also studied the trade-off between global and partial errors for those optimal values.

It was recalled that one can achieve an approximation error bound of $O(1/n)$, with $n$ sigmoidal units, obtained one at a time. From a practical standpoint we have revealed several potential barriers to achieving this bound:

1. The function $g_n$ in Step 2 of the IIA must be estimated from a finite number of examples.
2. The solution to Step 2 of the IIA has no closed form, and must be sought algorithmically.
3. No provably efficient algorithm is guaranteed to produce the optimal function at Step 2, or for that matter a function that is within the $O(1/n^2)$ tolerance.

In spite of these barriers, it is reasonable to assume that the $O(1/n)$ bound on approximation error can actually be achieved in practice, as long as the training examples are sufficiently rich. For higher dimensional problems it becomes essential to use algorithms like the one in Figure 8 if we hope to achieve good approximations in a reasonable time. In addition, we must often employ the refitting procedure in Figure 6 to compensate for the suboptimal functions produced in Step 2 of the IIA. Future work will concentrate on studying Step 2 of the IIA, and on studying approximating results for dynamical neural networks from a constructive point of view.

## 8 Acknowledgments

The research of D. Docampo was supported by CICYT (TIC96-0500-C10-10), ISTEC, and the US Air Force. The research of C.T. Abdallah and D.R. Hush was supported by ISTEC, and Boeing Computer Services Contract number W-30045.

## 9 References

[1] P. Auer, M. Herbster, and M.K. Warmuth. Exponentially many local minima for single neurons. In D. Touretzky, M.C. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 316–322. Morgan Kaufmann, 1996.

[2] P. Baldi. Gradient descent learning algorithm overview: A general dynamical systems perspective. *IEEE Trans. Neural Nets*, 6(1):182–195, 1995.

[3] A.R. Barron. Statistical properties of artificial neural networks. In *Proceedings of the 28th IEEE Conf. on Decision and Control*, pages 280–285, 1989.

[4] A.R. Barron. Approximation and estimation bounds for artificial neural networks. In L.G. Valiant and M.K. Warmuth, editors, *Proceedings of the 4th Annual Workshop on Computational Learning Theory*, pages 243–249, 1991.

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

[6] R. Battiti. First- and second–order methods for learning: between steepest descent and newton's method. *Neural Computation*, 4(2):141–166, 1992.

[7] L. Breiman and J.H. Friedman. Function approximation using ramps. In *Snowbird Workshop on Machines that Learn*, 1994.

[8] L. Breiman. Hinging hyperplanes for regression, classification and function approximation. *IEEE Trans. on Inf. Theory*, 39(3), 1993.

[9] T. Chen, H. Chen, and R-W. Liu. Approximation capability in C($\bar{R}^n$) by multilayer feedforward networks and related problems. *IEEE Trans. Neural Nets*, 6(1):25–30, 1995.

[10] E.W. Cheney. Topics in approximation theory, 1992.

[11] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989.

[12] 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.

[13] J.H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19, 1991.

[14] J.H. Friedman and W. Stuetzle. Projection pursuit regression. *J. Amer. Stat. Assoc.*, 76:817–823, 1981.

[15] J.H. Friedman, W. Stuetzle, and A. Schroeder. Projection pursuit density estimation. *J. Amer. Stat. Assoc.*, 79:599–608, 1984.

[16] J.H. Friedman and J.W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, C–23(9):881–890, 1974.

[17] F. Girosi and G. Anzellotti. Convergence rates of approximation by translates. Technical Report 1288, MIT Art. Intell. Lab., 1992.

[18] Simon Haykin. *Neural Networks: A Comprehensive Foundation.* Macmillan, New York, 1992.

[19] D. Hush, B. Horne, and J.M. Salas. Error surfaces for multi–layer perceptrons. *IEEE Transactions on Systems, Man and Cybernetics*, 22(5):1152–1160, 1992.

[20] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[21] P.J. Huber. Porjection pursuit. *The Annuals of Statistics*, 13(2):435–475, 1985.

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

[23] G. Pisier. Remarques sur un resultat non publié de b. maurey, 1980–1981.

[24] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, pages 318–362. MIT Press, Cambridge, MA, 1986.

[25] I.W. Sandberg. Structure theorems for nonlinear systems. *Multidim. Syst. and Sign. Proc.*, 2:267–286, 1991.

[26] I.W. Sandberg. Uniform approximation and the circle criterion. *IEEE Trans. Automat. Control*, 38(10):1450–1458, 1993.

[27] I.W. Sandberg. General structures for classification. *IEEE Trans. Circ. and Syst.–1*, 41(5):372–376, 1994.

[28] Y. Zhao. *On projection pursuit learning.* PhD thesis, Dept. Math. Art. Intell. Lab., MIT, Boston, MA, 1992.