III.3

Theory of the Backpropagation Neural Network*

ROBERT HECHT-NIELSEN

HNC, Inc.
and
University of California, San Diego

I Introduction

This paper presents a survey of some of the elementary theory of the basic backpropagation neural network architecture, covering the areas of: architectural design, performance measurement, function approximation capability, and learning. The survey includes previously known material, as well as some new results: a formulation of the backpropagation neural network architecture to make it a valid neural network (past formulations violated the locality of processing restriction) and a proof that the backpropagation mean squared error function exists and is differentiable. Also included is a theorem showing that any L_2 function from $[0,1]^n$ to R^m can be implemented to any desired degree of accuracy with a three-layer backpropagation neural network. Finally, an Appendix presents a speculative neurophysiological model illustrating how the backpropagation neural network architecture might plausibly be implemented in the mammalian brain for cortico-cortical learning between nearby regions of cerebral cortex. This paper is a slightly altered reprint of [25]. For a more comprehensive discussion of neural networks, the reader may find the author's graduate neurocomputing textbook [23] helpful.

*** Based on "Theory of the Backpropagation Neural Network" by Robert Hecht-Nielsen, which appeared in *Proceedings of the International Joint Conference on Neural Networks* 1, 593–611, June 1989. © 1989 IEEE.**

The backpropagation network has a colorful history. Apparently, it was originally introduced by Werbos in 1974 [65,62,63,64] (although Bryson and Ho published a mathematically similar concept in 1969 [6]) and independently rediscovered by Parker in the mid-1980's [50,48,49] and by Rumelhart, Williams and other members of the PDP group in 1985 [57,55,2]. Although the PDP group became aware of Parker's work shortly after their discovery (they cited Parker's 1985 report in their first papers on backpropagation [73,57]), Werbos' work was not widely appreciated until mid-1987. The work of Bryson and Ho was pointed out in 1988 by le Cun [43]. Even earlier incarnations may yet emerge.

Notwithstanding its checkered history, there is no question that credit for developing backpropagation into a usable technique, as well as for promulgation of the architecture to a large audience, rests entirely with Rumelhart, Williams, and the other members of the PDP group [56]. Before their work, backpropagation was unappreciated and obscure. Today, it is a mainstay of neurocomputing.

One of the crucial decisions in the design of the backpropagation architecture is the selection of a sigmoidal activation function (see Section 2 below) – although it is now known that other activation functions can be used [31,32,60]. Historically, sigmoidal activation functions have been used by a number of investigators. Grossberg [20] was probably the first advocate of the use of sigmoid functions in neural networks; although his reasons for using them are not closely related to their role in backpropagation (see [9] for a discussion of the relationship between these two bodies of work). Sejnowski, Hinton, and Ackley [27,56] and Hopfield [29] provided still other reasons for using sigmoidal activation functions, but again, these are not directly related to backpropagation. The choice of sigmoid activation for backpropagation (at least for the PDP group reincarnation of the architecture) was made quite consciously by Williams, based upon his careful 1983 study of activation functions [74].

Much of what appears in this paper concerns only the feedforward portion of the backpropagation network's operation. Thus, these results would apply to other feedforward networks (that might, for example, use a different learning law). The terms "feedforward network" and "multi-layer perceptron" are often used to denote networks such as that described in this paper when the learning law used is not being discussed.

II Backpropagation Neural Network Architecture

This Section reviews the architecture of the basic backpropagation neural network. The transfer function equations for each processing element are provided for both the forward and backward passes. First, we recall the definition of a neural network:

> **Definition:** A *neural network* is a parallel, distributed information processing structure consisting of *processing elements* (which can possess a local memory and can carry out localized information processing operations) interconnected together with unidirectional signal channels called *connections*. Each processing element has a single output connection which branches ("fans out") into as many collateral connections as desired (each carrying the same signal - the *processing element output signal*). The processing element output signal can be of any mathematical type desired. All of the processing that goes on within each processing element must be completely local: i.e., it must depend only upon the current values of the input signals arriving at the processing element via impinging connections and upon values stored in the processing element's local memory.

The importance of restating the neural network definition relates to the fact that (as pointed out by Carpenter and Grossberg [9], and by Crick [11]) traditional forms of the backpropagation architecture are, in fact, not neural networks. They violate the locality of processing restriction. The new backpropagation neural network architecture presented below eliminates this objection, while retaining the traditional mathematical form of the architecture.

The backpropagation neural network architecture is a hierarchical design consisting of fully interconnected *layers* or *rows* of processing *units* (with each unit itself comprised of several individual processing elements, as will be explained below). Backpropagation belongs to the class of *mapping neural network* architectures and therefore the information processing function that it carries out is the approximation of a bounded mapping or function $f : A \subset \mathbf{R}^n \longrightarrow \mathbf{R}^m$, from a compact subset A of n-dimensional Euclidean space to a bounded subset $f[A]$ of m-dimensional Euclidean space, by means of training on examples $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_k, \mathbf{y}_k), \ldots$ of the mapping, where $\mathbf{y}_k = f(\mathbf{x}_k)$. It will always be assumed that such examples of a mapping f are generated by selecting $\mathbf{x}_k$ vectors randomly from A in accordance with a fixed probability density function $\rho(\mathbf{x})$. The operational use to which the network is to be put after training is also assumed to involve random selections of input vectors $\mathbf{x}$ in accordance with $\rho(\mathbf{x})$. The backpropagation architecture described in this paper is the basic, classical version. A large number of variants of this basic form exist (see Section 5).

The macro-scale detail of the backpropagation neural network architecture is shown in Figure 1. In general, the architecture consists of K rows of processing units, numbered from the bottom up beginning with 1. For simplicity, the terms *row* and *layer* will be used interchangeably in this paper, even though each row will actually turn out to consist of two heterogeneous layers (where the term *layer* is used to denote a collection of processing elements having the same form of transfer function). The first layer consists of n fanout processing elements that simply accept the individual components x_i of the input vector $\mathbf{x}$ and distribute them, without modification, to all of the units of the second row. Each unit on each row receives the output signal of each of the units of the row below. This continues through all of the rows of the network until the final row. The final (K^{th}) row of the network consists of m units and produces the network's estimate $\mathbf{y}'$ of the correct output vector $\mathbf{y}$. For the purposes of this paper it will always be assumed that $K \geq 3$. Rows 2 thru $K - 1$ are called *hidden* rows (because they are not directly connected to the outside world).

Besides the feedforward connections mentioned above, each unit of each hidden row receives an 'error feedback' connection from each of the units on the next row. However, as will be seen below, these are not merely fanned out copies of a broadcast output (as the forward connections are), but are each separate connections, each carrying a different signal. The details of the individual "units" (shown as rectangles in Figure 1) are revealed in Figure 2 (which depicts two units on adjacent rows and shows all of their connections – except those from the scheduling element, see below). Note that each unit is composed of a single *sun* processing element and several *planet* processing elements. Each planet produces an output signal that is distributed to both its sun and to the sun of the previous layer that supplied input to it. Each planet receives input from one of the suns of the previous layer as well as its own sun. As stated above, the hidden row suns receive input from one of the planets of each of the suns on the next higher row. The output row suns receive the 'correct answer' y_i for their component of the output vector on each training trial. As discussed in detail below, the network functions in two stages: a forward pass and a backward pass. A *scheduling processing element* (not shown) sends signals to each of the processing elements of the network telling it when to apply its processing element transfer function and whether to apply the forward pass part of it or the backward pass part of it. After the transfer function is applied, the output signal is latched to the value determined during the update. This value is therefore constant until the next update. The exact transfer functions of the processing elements of the network are given in Table 1.

The scheduling of the network's operation consists of two "sweeps" through the network. The first sweep (the *forward pass*) starts by inserting the vector $\mathbf{x}_k$ into the network's first row, the *input* (or *fanout*) layer. The processing elements of the first layer have no function other than to transmit all of the components of $\mathbf{x}_k$ to all of the units of the second row of the network. The outputs of the units of row two are then transmitted to all of the units of row three, and so on, until finally the *m output units* (the units of the top, K^{th}, row) emit the components of the vector $\mathbf{y}'_k$ (the network's estimate of the desired output $\mathbf{y}_k$). After the estimate $\mathbf{y}'_k$ is emitted, each of the output units is supplied with its component of the correct output vector $\mathbf{y}_k$, starting the second, backward, sweep through the network (the *backward pass*). The output suns compute their δ_{Ki}'s and transmit these to their planets. The planets then update their Δ_{Kij} values (or update their weights) and then transmit the values $w^{old}_{Kij}\delta_{Ki}$ to the suns of the previous row. This weight modification law is radically non-Hebbian. The superscript "old" indicates that the (non-updated) weight value used on the forward pass is used in this calculation. This process continues until the planets of row 2 (the first hidden layer) have been updated. The cycle can then be repeated. In short, each cycle consists of the inputs to the network 'bubbling up' from the bottom to the top and then the errors 'percolating down' from the top to the bottom. This process is continued until the network reaches a satisfactory level of performance (which is tested using data that was not used during training, but which was drawn at random in the same way as the training data); or until the user gives up and decides to try a new starting weight set or a new network configuration (see Section 5 below for a discussion of backpropagation network training). After successfully training the network to a suitably low level of error the network can then be put through further operational testing to qualify it for deployment. Since training is typically not used in the deployed version, the backward pass can be eliminated – allowing a considerable computational savings to be realized.

III Backpropagation Error Surfaces

Given a function f, an associated $\mathbf{x}$-selection probability density function ρ, and an associated backpropagation architecture (i.e., with a given number of layers K and a given number of units M_l per hidden layer) intended to approximate f, we now define a means for measuring the accuracy of

Forward Pass	**Backward Pass**
Planet j of sun i of row l ($l = 2, 3, \ldots, K$): Input Used: $z_{(l-1)j}$ (where $z_{(l-1)0} \equiv 1.0$) Weight Value Used: w_{lij} Local Memory Value Used: None Output: $w_{lij}\ z_{(l-1)j}$ Weight and Local Memory Value Update and Storage: None	Input Used: z_{li} ($= \delta_{li}$) Weight Value Used: w_{lij} Local Memory Values Used: count, Δ_{lij} Output: $w^{old}_{lij}\ \delta_{li}$ Weight and Local Memory Value Update and Storage: IF (count = batch_size) THEN { $w^{new}_{lij} = w^{old}_{lij} + \alpha\Delta_{li}/\text{batch\_size}$ $\Delta^{new}_{lij} = \delta_{li} z_{(l-1)j}$ count = 1 } ELSE { $w^{new}_{lij} = w^{old}_{lij}$ $\Delta^{new}_{lij} = \Delta^{old}_{lij} + \delta_{li} z_{(l-1)j}$ count = count + 1 }
Hidden Sun i of row l ($l = 2, 3, \ldots, K-1$): Inputs Used: $w_{li0}z_{(l-1)0}, w_{li1}z_{(l-1)1}, \ldots, w_{liM_{(l-1)}}z_{(l-1)M_{(l-1)}}$ Local Memory Value Used: None Output: $z_{li} = s(\sum_j w_{lij}z_{(l-1)j})$ Local Memory Value Stored: $I_{li} \equiv \sum_j w_{lij}z_{(l-1)j}$	Inputs Used: $w_{(l+1)1i}\delta_{(l+1)1}$, $w_{(l+1)2i}\delta_{(l+1)2}, \ldots,$ $w_{(l+1)M_{(l+1)}i}\delta_{(l+1)M_{(l+1)}}$ Local Memory Value Used: I_{li} Output: $z_{li} = \delta_{li}$ $\equiv s'(I_{li})\sum_k w_{(l+1)ki}\delta_{(l+1)k}$ Local Memory Value Stored: None
Output Sun i of row l = K: Inputs Used: $w_{Ki0}z_{(K-1)0}, w_{Ki1}z_{(K-1)1}, \ldots, w_{KiM_{(K-1)}}z_{(K-1)M_{(K-1)}}$ Local Memory Value Used: None Output: $y'_i = z_{Ki} = \sum_j w_{Kij}z_{(K-1)j}$ Local Memory Value Stored: y'_i	Input Used: y_i Local Memory Value Used: None Output: $z_{Ki} = \delta_{Ki}$ $\equiv (y_i - y'_i)$ Local Memory Value Stored: None

where $s(I) = 1/(1+e^{-I})$ is the ***sigmoid function*** of the network, $s' = ds/dI$ is the first derivative of the sigmoid function, M_l is the number of units on row l, w_{lij} is the weight of planet j of unit i of row l, $z_{1i} = x_i$ (where x_i is the i^{th} component of the input vector $\mathbf{x}$), and the network's output signals z_{Ki} are equal to the components y'_i of the network's output vector $\mathbf{y}'$ – the network's estimate of the 'correct' or 'desired' output $\mathbf{y}$ (the components y_i of which are supplied to the network during each training trial).

Table 1: Processing element transfer functions for the backpropagation neural network architecture. Three types of processing elements are used (planets, hidden suns, and output suns). For each processing element type the transfer function used on the forward pass of the network and the backward pass of the network is given. The backward pass occurs only on training trials.

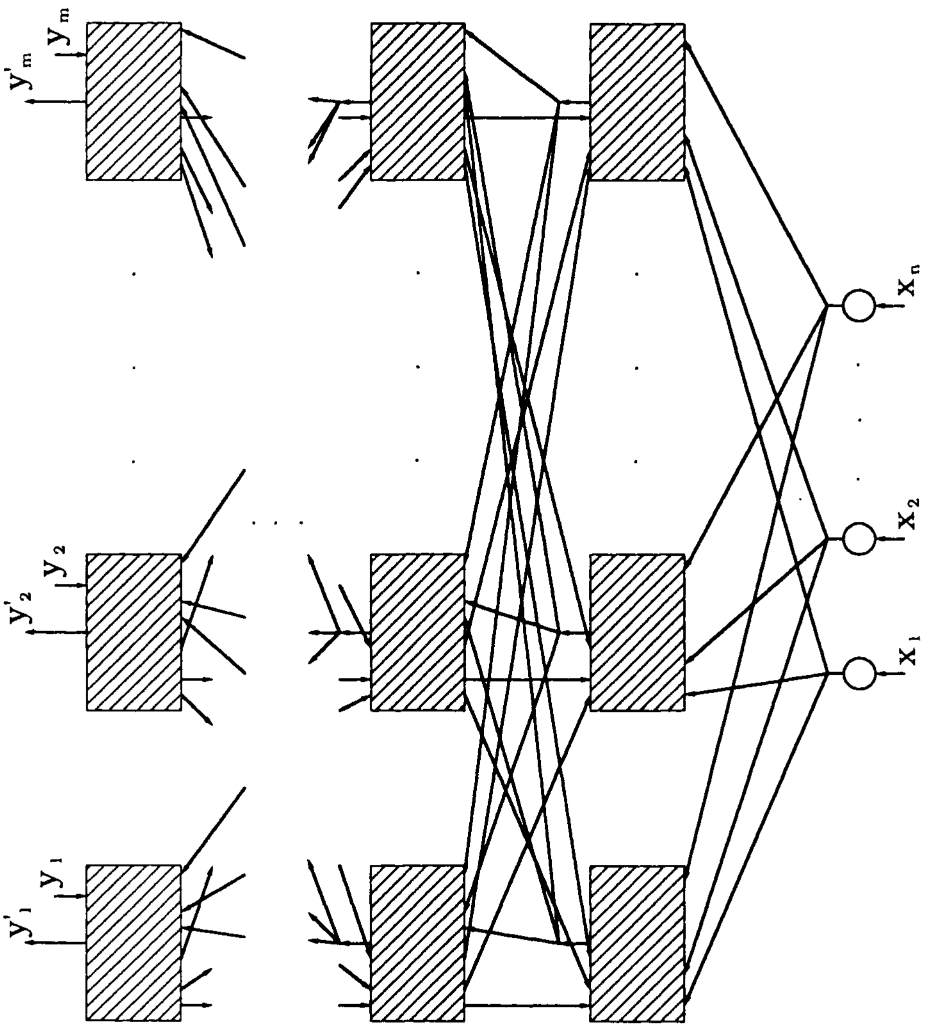

Figure 1: Macroscopic Architecture of the Backpropagation Neural Network. The boxes are called *units*. The detailed architecture of the units is elaborated in Figure 2. Each row can have any number of units desired; except the output row, which must have exactly m units (because the output vector $\mathbf{y}'$ must be m-dimensional).

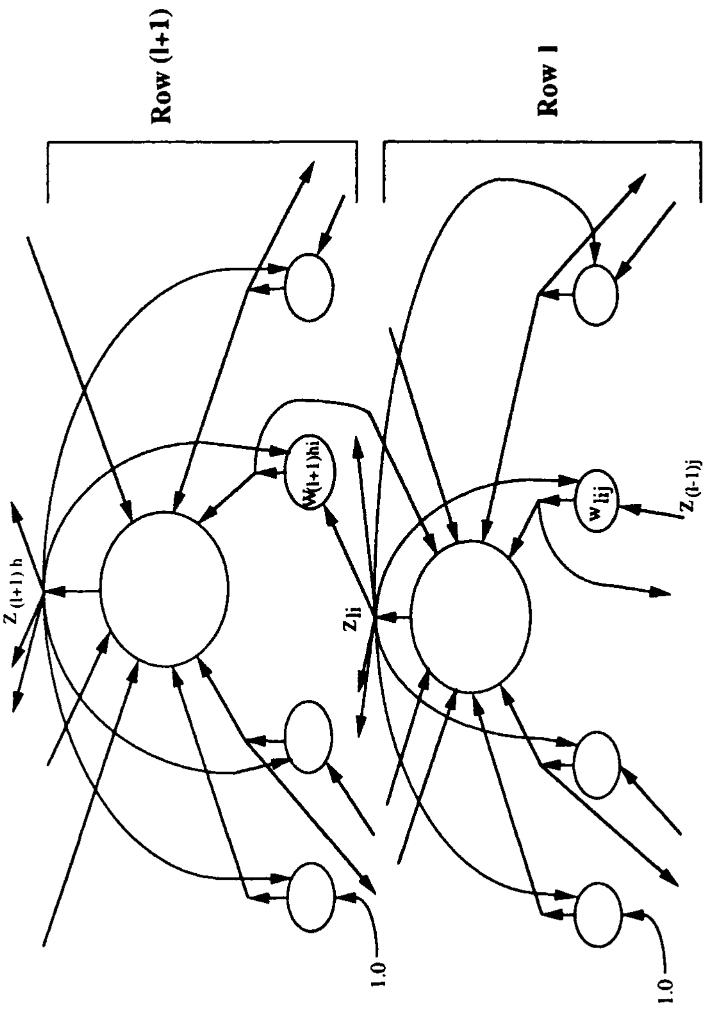

Figure 2: Architectural detail and interaction of two backpropagation network units on two adjacent rows (row l below and row $l+1$ above). Each unit consists of a *sun* and several *planets*. Each hidden row sun sends its output to one planet of each unit of the next row. Each planet that receives an input from the previous row (except the *bias* planets, which all receive the constant input $z_{(l-1)0} \equiv 1.0$ from a single processing element which is not shown) sends a connection back to the sun that supplied that input. It is through these paths that the products of the errors times the weights $(w_{lij}\delta_{li})$ are back-propagated.

this approximation as a function of the network's weights and discuss the resulting *error surface*.

First, define $\mathbf{w}$ (the *weight vector* of the network) to be the vector with components consisting of the weights of all of the planets of the network, starting with the weight of the first planet of the first processing element of layer 2 (the first hidden layer) and ending with the weight of the last planet of the last processing element of the output layer K. To make things simple we shall refer to the components of $\mathbf{w}$ as $w_1, w_2, \ldots$; rather than $w_{210}, w_{211}, \ldots$. To make our notation simple the vector $(z_{K1}, z_{K2}, \ldots, z_{Km})$ (the network's estimate $\mathbf{y}'$ of the correct output $\mathbf{y}$) shall be called $\mathbf{B}$. Note that $\mathbf{B}$ is a function of the input vector $\mathbf{x}$ and the network weight vector $\mathbf{w}$. Thus, we will write $\mathbf{B}(\mathbf{x}, \mathbf{w})$. Let $(\mathbf{x}_k, \mathbf{y}_k)$ be the example used on the k^{th} testing trial (i.e., $\mathbf{y}_k = f(\mathbf{x}_k)$). As before, the $\mathbf{x}_k$'s are drawn from A in accordance with a fixed probability density function ρ.

Given the above, let $F_k = |f(\mathbf{x}_k) - \mathbf{B}(\mathbf{x}_k, \mathbf{w})|^2$. F_k is the square of the approximation error made on the k^{th} testing trial. For the purposes of the discussion below we shall assume that $\mathbf{w}$ is fixed (i.e., we will only be doing forward passes during testing). Then we define $F(\mathbf{w})$ to be

$$F(\mathbf{w}) \equiv \lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} F_k. \tag{1}$$

Because the mapping f is assumed bounded the set of all $\mathbf{y}_k$'s are bounded. Thus, it is easy to show that this limit exists because (for a fixed $\mathbf{w}$) $\mathbf{B}$ is a continuous mapping from the compact set A into $\mathbf{R}^m$, and thus the set of all $\mathbf{B}$'s are bounded also. Thus, the variance of the random variable F_k is bounded. So, by Kolmogorov's probability theorem [5], the random variable F_k obeys the strong law of large numbers. Therefore, the above F sum must almost surely converge to the expected value $F(\mathbf{w})$ of F_k. We call $F(\mathbf{w})$ the *mean squared error* function of the network; which is often just shortened to *error function*. Note that $F(\mathbf{w}) \geq 0$ because F is the average of non-negative quantities.

The *error surface* of a backpropagation network is the surface defined by the equation $F = F(\mathbf{w})$ in the $Q+1$-dimensional space of vectors $(\mathbf{w}, F)$, where Q is the number of dimensions in the vector $\mathbf{w}$ (i.e., the number of planets in the network). The variable $\mathbf{w}$ ranges over its Q-dimensional space and for each $\mathbf{w}$ a non-negative surface height F is defined by $F(\mathbf{w})$. In other words, given any selection of weights $\mathbf{w}$, the network will make an average squared error $F(\mathbf{w})$ in its approximation of the function f. We now consider the shape of this error surface.

As will be shown below, the generalized delta rule learning law used with the backpropagation neural network has the property that, given any starting point $\mathbf{w}_0$ on the error surface that is not a minimum, the learning law will (usually if the "learning rate" α is sufficiently small) modify the weight vector $\mathbf{w}$ so that $F(\mathbf{w})$ will decrease. In other words, the learning law uses examples provided during training to decide how to modify the weight vector so that the network will do a better job of approximating the function f. Given this behavior, the next issue is to assess how valuable this property is.

Until recently, the shape of backpropagation error surfaces was largely a mystery. Two basic facts have emerged so far. First, experience has shown that many backpropagation error surfaces are dominated by flat areas and troughs that have very little slope. In these areas it is necessary to move the weight value a considerable distance before a significant drop in error occurs. Since the slope is shallow it turns out that the generalized delta rule has a hard time determining which way to move the weight to reduce the error. Often, great numerical precision (e.g., 32-bit floating point) and patience must be employed to make significant progress.

The other basic fact about error surfaces that has emerged concerns the existence of local minima. Until recently, it was not known for certain whether backpropagation error surfaces have local minima at error levels above the levels of the global minima of the surfaces (due to weight permutations there are always many global minima). Experience suggested that such minima might not exist because usually when training failed to make downhill progress and the error level was high it was discovered that further patience (or the use of one of the training augmentations alluded to in Section 5 below) would eventually lead to the weight moving away from what was clearly a shallow spot on the surface and onto a steeper part. Thus, it was somewhat surprising when McInerney, Haines, Biafore, and the author [45] discovered a local minimum (at a very high error level) in a backpropagation error surface in June 1988. Finding this local minimum was not easy. It required the use of the Symbolics Corporation's Macsyma$^{\text{TM}}$ mathematics expert system running on the largest Digital Equipment Corporation VAX$^{\text{TM}}$ computer to generate approximately a megabyte of FORTRAN code (the closed-form formulas for the error function and all of its first and second partial derivatives with respect to the weights) and a 12-hour run on a Cray-2 supercomputer (of a program that called this error surface code) to accomplish this discovery. The existence of the minimum was proven by showing that all of the first partial derivatives of the mean squared error function went to zero at this point and that the Hessian of this function (the matrix of second partial derivatives) was strongly positive-definite at this point.

In summary, some basic facts are known about backpropagation error surfaces. First, because of combinatorial permutations of the weights that leave the network input-output function unchanged, these functions typically have huge numbers of global minima (which may lie at infinity for some problems). This causes the error surfaces to be highly degenerate and to have numerous 'troughs'. Secondly, error surfaces have a multitude of areas with shallow slopes in multiple dimensions simultaneously. These typically occur because particular combinations of weights cause the weighted sums of one or more suns (with sigmoided outputs) to be large in magnitude. When this occurs the output of that sun (and therefore the value of F) is insensitive to small weight changes, since these simply move the weighted sum value back and forth along one of the shallow tails of the sigmoid function. Thirdly, it is now established that local minima do actually exist. Beyond these three facts, little is known. How many non-global minima are there, compared to the number of global minima? Are local minima excluded from regions near global minima? How large are the attractive basins of each of the different types of minima? What forms do the boundaries between attractive basins take? Can some of the nonlinear optimization and unconstrained optimization techniques developed by researchers in operations research and mathematical economics be usefully applied? Clearly, more research into error surfaces is needed (see [22,52,24] for some recent results).

IV Function Approximation with Backpropagation

The question of which functional forms can be approximated by neural networks has had pivotal importance in the history of neurocomputing. For example, the fact that a narrowly formulated type of perceptron could be shown incapable of implementing the EXCLUSIVE OR logical operation [47] was used in the 1960's as an argument to divert funding from neurocomputing to artificial intelligence. Recently, similar questions have been raised and claims that little progress has been made on this front [47] have generated concern.

Clear insight into the versatility of neural networks for use in function approximation came with the discovery [26] that a classic mathematical result of Kolmogorov [38] was actually a statement that for any continuous mapping $f : [0,1]^n \subset \mathbf{R}^n \longrightarrow \mathbf{R}^m$ there must exist a three-layer neural network (having an input or "fanout" layer with n processing elements, a hidden layer with $(2n+1)$ processing elements, and an output layer with m

processing elements) that implements f *exactly*. This result gave hope that neural networks would turn out to be able to approximate any function that arises in the real world.

Kolmogorov's theorem was a first step. The following result shows that the backpropagation network is itself able to implement any function of practical interest to any desired degree of accuracy. To make the exposition precise, we start with some background information.

Let $[0,1]^n$ be the closed unit cube in n-dimensional Euclidean space. Given any square-integrable function $g : [0,1]^n \subset \mathbf{R}^n \longrightarrow \mathbf{R}$ (i.e., $\int_{[0,1]^n} |g(\mathbf{x})|^2 d\mathbf{x}$ exists), it can be shown by the theory of Fourier series [14] that the series

$$\begin{aligned}\hat{g}(\mathbf{x}, N) &= \sum_{k_1=-N}^{N} \sum_{k_2=-N}^{N} \cdots \sum_{k_n=-N}^{N} c_{k_1 k_2 \dots k_n} \exp\left(2\pi i \sum_{q=1}^{n} k_q x_q\right) \\ &= \sum_{\mathbf{k}} c_{\mathbf{k}} \exp(2\pi i \ \mathbf{k} \cdot \mathbf{x})\end{aligned} \qquad (2)$$

where

$$c_{k_1 k_2 \dots k_n} = c_{\mathbf{k}} = \int_{[0,1]^n} g(\mathbf{x}) \ \exp(-2\pi i \ \mathbf{k} \cdot \mathbf{x}) \ d\mathbf{x} \qquad (3)$$

converges to g in the sense that

$$\lim_{N \to \infty} \int_{[0,1]^n} |g(\mathbf{x}) - \hat{g}(\mathbf{x}, N)|^2 \, d\mathbf{x} = 0. \qquad (4)$$

This is an example of the property of having the integral of the square of the error of approximation of one function by another go to zero. This property is described by the statement that the approximation can be achieved to within any desired degree of accuracy in the *mean squared error sense*. This leads to a definition.

Given a function $f : [0,1]^n \subset \mathbf{R}^n \longrightarrow \mathbf{R}^m$, we say that f *belongs to* L_2 (or "*is* L_2") if each of f's coordinate functions is square-integrable on the unit cube. For functions of this class it is assumed that the $\mathbf{x}$ vectors are chosen uniformly in $[0,1]^n$ (this condition can be relaxed). Clearly if a vector function of a vector variable is in L_2 then each of its components can be approximated by its Fourier series to any desired degree of accuracy in the mean squared error sense. With this background as preamble, the new result is now presented.

Theorem 1 : *Given any $\epsilon > 0$ and any L_2 function*

$$f : [0,1]^n \subset \mathbf{R}^n \longrightarrow \mathbf{R}^m,$$

there exists a three-layer backpropagation neural network that can approximate f to within ϵ mean squared error accuracy.

Proof of Theorem 1: Let ϵ/n be the accuracy to which we wish to approximate each of the coordinate functions f_l (where $f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x}))$ of f by using a three-layer backpropagation neural network. Note that, by virtue of the results from Fourier series theory cited above, given any $\delta_1 > 0$, there exists a positive integer N and coefficients $c_{l\mathbf{k}}$ such that

$$\int_{[0,1]^n} |f_l(\mathbf{x}) - \sum_{\mathbf{k}} c_{l\mathbf{k}} \exp(2\pi i \mathbf{k} \cdot \mathbf{x})\,|^2 d\mathbf{x} < \delta_1. \tag{5}$$

We begin by showing that each of the sine and cosine terms required in the Fourier series (i.e., in the real part of the complex Fourier series – the imaginary part vanishes) for f_l can be implemented to any desired degree of absolute accuracy by a subset of a three-layer backpropagation neural network. The idea is to use the input layer units, a contiguous subset of the hidden layer units, and the corresponding portion of a single output layer unit – unit l out of a total of m output units – to implement each sine or cosine function. To carry out this approximation, first note that the input layer, a subset H of the hidden layer (assumed here to be comprised of contiguous hidden units), and the l^{th} output unit can compute any sum of the form

$$\sum_{i \in H} v_{li} \; s\left(\sum_{j=0}^{n} w_{ij} x_j\right) \tag{6}$$

where $x_0 \equiv 1$ is the bias input to each hidden unit. Next, we note that each of the arguments of the sine and cosine functions in the Fourier approximation of f_l (namely the terms $u = u(\mathbf{k}, \mathbf{x}) = 2\pi i\ \mathbf{k} \cdot \mathbf{x}$) are of the form

$$\sum_{j=0}^{n} w_{ij} x_j. \tag{7}$$

Since, by simply adjusting the argument bias weight w_{i0} by $-(\pi/2)$ we can change a sine into a cosine, we shall concern ourselves exclusively with sines. Thus, to absolutely approximate a particular Fourier series sine or cosine function all we need to show is that, given any $\delta_2 > 0$, we can find coefficients v_{li} and w_{ij} such that

$$|\sin(u(\mathbf{k},\mathbf{x})) - \sum_{i \in H} v_{li} \; s\left(\sum_{j=0}^{n} w_{ij} x_j\right) | < \delta_2 \tag{8}$$

for each $\mathbf{x} \in [0,1]^n$. To show this, choose the w_{ij} such that

$$\sum_{j=0}^{n} w_{ij} x_j = \beta_i \; (u(\mathbf{k},\mathbf{x}) - \alpha_i) \tag{9}$$

where the quantities β_i and α_i are arbitrary real constants to be selected below. We are going to dedicate a certain (typically large) subset $H_{\mathbf{k}}$ of hidden layer units (assumed to be contiguous) to the calculation of each $\sin(u(\mathbf{k},\mathbf{x}))$ and $\cos(u(\mathbf{k},\mathbf{x}))$. So, with this transformation for each hidden layer unit within $H_{\mathbf{k}}$ we can rewrite the above inequality as

$$|\sin(u(\mathbf{k},\mathbf{x})) - \sum_{i \in H_{\mathbf{k}}} v_{li} \; s\,(\beta_i(u(\mathbf{k},\mathbf{x}) - \alpha_i))\,| < \delta_2 \tag{10}$$

It is easy to show, given a sufficient number of hidden layer units in $H_{\mathbf{k}}$, that this inequality can always be satisfied. The first step in this demonstration is to examine the geometrical form of the sum

$$S(\alpha,\beta,\mathbf{v}_l,\mathbf{x}) \equiv \sum_{i \in H_{\mathbf{k}}} v_{li} \; s\,(\beta_i\,(u - \alpha_i)) \tag{11}$$

which, by the above argument, can be computed by the input layer, hidden layer subset $H_{\mathbf{k}}$, and output unit l. Note that for $\mathbf{x} \in [0,1]^n$, u ranges over some closed interval $d(\mathbf{k}) \leq u \leq e(\mathbf{k})$. It is a closed interval because the affine transformation $\mathbf{x} \mapsto u$ is continuous and continuous maps preserve compactness and simple connectedness. So, we need to approximate the function $\sin(u)$ on the closed interval $d \leq u \leq e$ using this sum. To do this, partition the interval as follows

$$d = \alpha_{i_1} < \alpha_{i_2} < \alpha_{i_3} < \cdots < \alpha_{i_{last}} = e \tag{12}$$

where $i_{p+1} = i_p + 1$ and $\cup_{p=1}^{last}\{i_p\} =$ index set of $H_{\mathbf{k}}$.

By letting $\beta_{i_1} = 0$ and by choosing the β_{i_p}, $p > 1$ to be sufficiently large it is easy to see that the sum $S(\alpha,\beta,\mathbf{v}_l,\mathbf{x})$ has the geometrical form shown in Figure 3, since each of the terms in the sum is a steep sigmoid (sort of like a step function with microscopically rounded corners) that is essentially equal to 1 for $u > \alpha_{i_p}$ and equal to 0 for $u < \alpha_{i_p}$ (and equal to 0.5 when $u = \alpha_{i_p}$).

Clearly, S has the approximate form of a staircase, where the step widths are determined by the differences $(\alpha_{i_{p+1}} - \alpha_{i_p})$ and the step heights are determined by the coefficients v_{li_p}. By setting the sigmoid gains β_{i_p}, $p > 1$ to high enough values this basic staircase form can always be achieved, no matter how small the steps become or how many steps are used.

Given the above facts about the geometrical form of the sum S, Figure 4 demonstrates graphically that no matter how small $\delta_2 > 0$ is chosen, a sum $S(\alpha, \beta, \mathbf{v}_l, \mathbf{x})$ can be constructed so that it always remains within the δ_2 absolute error band around $\sin(u)$. By starting at the left end of the interval $[d, e]$, and working to the right, successive sigmoid stairsteps can be incrementally added to achieve this.

Thus, we have shown that by choosing $H_{\mathbf{k}}$ to have an adequately large number of processing elements and by properly selecting the α, β, and $\mathbf{v}_l$ vectors, the partial sum output (due to the members of $H_{\mathbf{k}}$) will be within δ_2 of $\sin(u(\mathbf{k}, \mathbf{x}))$ (or $\cos(u(\mathbf{k}, \mathbf{x})$, if that is what is being fitted), for all $\mathbf{x} \in [0,1]^n$. Since output unit l receives the inputs from the hidden layer units of each of the sine and cosine subsets $H_{\mathbf{k}}$ for *every* $\mathbf{k} \in V \equiv \{-N, -N+1, \ldots, N\}^n$ we can then approximately generate the Fourier series for f_l by simply multiplying all of the sine (or cosine) coefficients v_{li} for $i \in H_{\mathbf{k}}$ by the appropriate combinations of the real and imaginary parts of $c_{l\mathbf{k}}$ (the complex Fourier expansion coefficient of f_l for term $\mathbf{k}$ in the expansion). Call these sine and cosine multipliers $a(l, \mathbf{k})$ and $b(l, \mathbf{k})$, respectively, and let $y_l(\mathbf{x})$ be the output signal of output unit l. Thus, we get

$$\begin{aligned} \hat{f}_l(\mathbf{x}, N) - y_l'(\mathbf{x}) &= \textstyle\sum_{\mathbf{k} \in V} a(l, \mathbf{k})[\sin(2\pi i\ \mathbf{k} \cdot \mathbf{x}) - S(\alpha, \beta, \mathbf{v}_l, \mathbf{x})] \\ &\quad + b(l, \mathbf{k})[\cos(2\pi i\ \mathbf{k} \cdot \mathbf{x}) - S'(\alpha, \beta, \mathbf{v}_l, \mathbf{x})], \end{aligned} \tag{13}$$

where the S' sum is that used for the $H_{\mathbf{k}}$ cosine term. Putting all of the above together we get:

$$\begin{aligned} \int_{[0,1]^n} & |f_l(\mathbf{x}) - y_l'(\mathbf{x})|^2 dx \\ &= \int_{[0,1]^n} |f_l(\mathbf{x}) - \hat{f}_l(\mathbf{x}, N) + \hat{f}_l(\mathbf{x}, N) - y_l'(\mathbf{x})|^2 dx \\ &\le \left[\sqrt{\int_{[0,1]^n} |f_l(\mathbf{x}) - \hat{f}_l(\mathbf{x}, N)|^2 dx} + \sqrt{\int_{[0,1]^n} |\hat{f}_l(\mathbf{x}, N) - y_l'(\mathbf{x})|^2 dx}\right]^2 \\ &< \left[\sqrt{\delta_1} + \sqrt{(\delta_2)^2 \textstyle\sum_{\mathbf{k} \in V} ([a(l, \mathbf{k})]^2 + [b(l, \mathbf{k})]^2)}\right]^2 . \end{aligned} \tag{14}$$

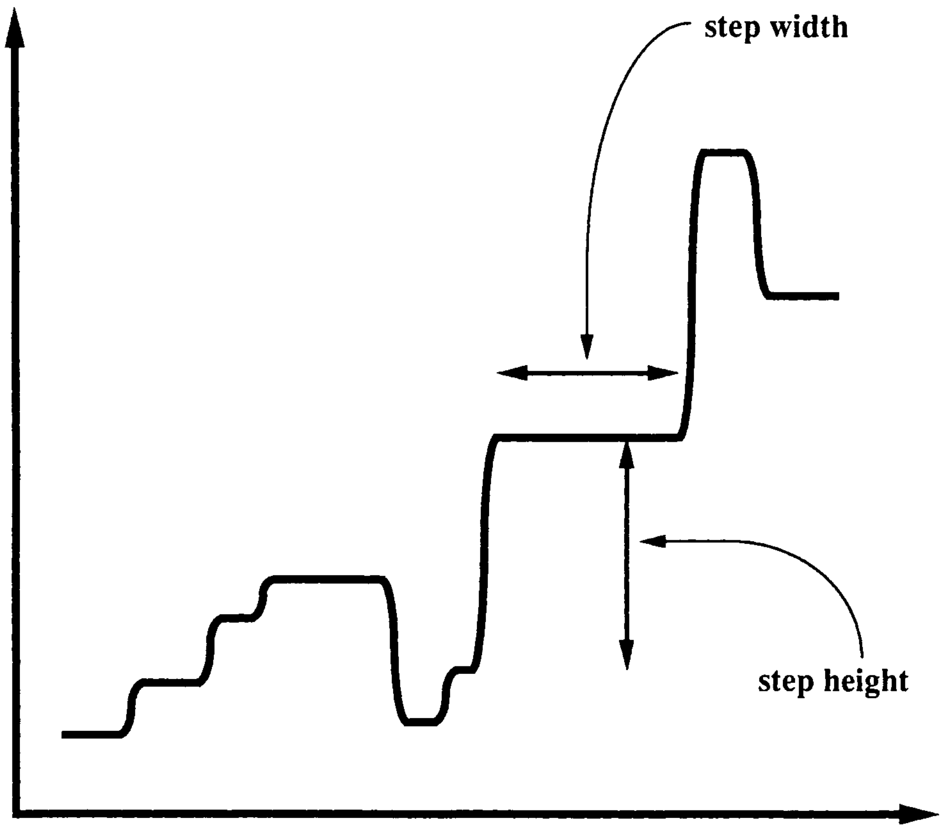

Figure 3: Geometrical Form of the function $S(\alpha, \beta, \mathbf{v}_l, \mathbf{x})$ with $\beta_{i_1} = 0$ and large β_{i_p} for $p > 1$.

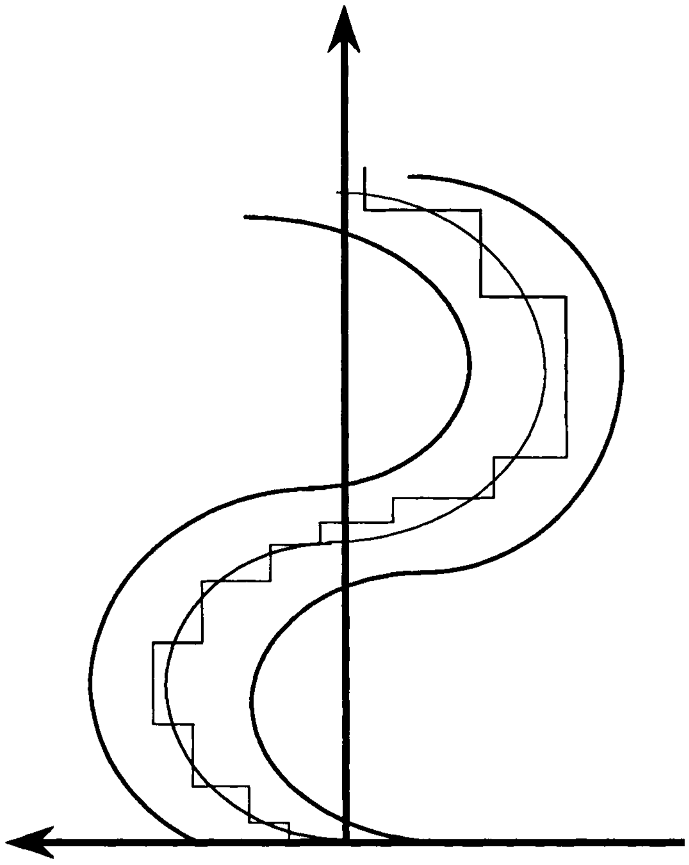

Figure 4: Graphical demonstration that α, β, and $\mathbf{v}_l$ vectors can always be chosen so that the function $S(\alpha, \beta, \mathbf{v}_l, \mathbf{x})$ will always remain within an absolute error band of width δ_2 around $\sin(u)$.

or

$$w_{lij}^{\mathrm{new}} = w_{lij}^{\mathrm{old}} - \alpha \lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} \delta_{li}^{k} \, z_{(l-1)j}^{k} \tag{34}$$

where $\alpha > 0$ is a small constant called the *learning rate*. This is the *generalized delta rule* learning law.

The rigorous generalized delta rule is approximated by the learning law used in the backpropagation architecture presented above, which substitutes a finite averaging sum of *batch_size* terms for the infinite-size average of the formula. Since we know that the limit converges, this *batching* version of the learning law will clearly be acceptable, assuming *batch_size* is chosen large enough.

A number of variants of this learning law (and of other parts of the backpropagation neural network architecture) have been presented [10,33, 59]. One learning law variant in common use is

$$w_{lij}^{\mathrm{new}} = w_{lij}^{\mathrm{old}} - \alpha \, \delta_{li}^{k} \, z_{(l-1)j}^{k}. \tag{35}$$

This *jump every time* variant of the generalized delta rule is the backpropagation analog of Widrow/Hoff learning. The proof that this law also carries out gradient descent on the backpropagation error surface was provided by Morris Hirsch [28] (see [66] for further details). Another common variant is the *momentum* version of the above law (see [23] for details).

The above learning laws suffer from the same basic problem: they move downhill in short jumps. Thus, even if the network's initial weight vector lies within the "attractive basin" of a global minimum of the error surface, getting to the bottom can take many jumps. Further, the magnitude of the gradient vector gets small when a shallow area is reached – thus *shortening* the jumps that are taken – which is exactly the reverse of what is needed.

Many other backpropagation network learning laws have been developed. The general goal is to provide a faster descent to the bottom of the error surface. One line of investigation [61,53,4] is exploring the use of approximations to the pseudoinverse of the Hessian matrix $H = [(\partial^2 F/\partial w_i \partial w_j)]$ to calculate individually variable learning rates (α values in Equation (34)) for each weight w_{lij}. The underlying concept is to use Newton's method for finding a place where $\nabla_{\mathbf{w}} F(\mathbf{w}) = 0$. This method expands the error function $F(\mathbf{w})$ in a Taylor series and then truncates this series after the third term to yield $F(\mathbf{w}) \doteq F(\mathbf{w}_0) + [\nabla_{\mathbf{w}} F(\mathbf{w})]|_{\mathbf{w}_0}(\mathbf{w} - \mathbf{w}_0) + (\mathbf{w} - \mathbf{w}_0)^{\mathrm{T}} H|_{\mathbf{w}_0}(\mathbf{w} - \mathbf{w}_0)$, where, again, H is the Hessian. Setting the gradient of this quantity to zero and solving for $\mathbf{w}^{\mathrm{new}}$ yields $\mathbf{w}^{\mathrm{new}} = \mathbf{w}_0 - H^{+}\nabla F$. Under some circumstances, the new value of $\mathbf{w}$ will be a very intelligent jump in the right direction, leading to a large decrease in F. In other situations, it does not work so well. Calculating H^+ is difficult. So far, this work shows promise, but a major advance in convergence speed has yet to be realized for arbitrary problems. Another approach that shows promise in early tests is the incremental network growing technique of Timur Ash [3]. This method starts with a small number of units in each hidden layer and adds additional units in response to changes in the error level (as measured using test data). Many other methods have been proposed, many of which provide a significant increase in descent speed over the laws discussed here (see [59] for a discussion of a set of these improved laws).

It is probably reasonable to anticipate that faster learning techniques will be developed in the future as more becomes known about the structure of backpropagation error surfaces, and as these facts are exploited. The advent of such faster learning techniques will correspondingly increase the breadth of applicability of the backpropagation architecture.

Finally, it is important to note that this section has examined only one (albeit perhaps the most important) variant of the backpropagation neural network architecture. Other important variants exist. Examples of these include architectures in which connections skip layers [56], recurrent architectures [56,51,70], and the 'sigma-pi' higher order architectures [56].

Methods for adding weight-dependent terms to a network's error function that create a force pushing the network towards a smaller number of non-zero weights, a smaller number of processing elements, or some other "goal" have also been demonstrated [64,54]. As these on-line methods and related off-line methods are developed further, it may become possible for backpropagation architectures to become "self-adjusting". Statistical tests for determining whether a backpropagation network is "overfitting" or "underfitting" a particular training set also exist (see [23] for details). These may someday allow the development of automated systems that will optimally craft a backpropagation network for a particular data set.

V Conclusions

As demonstrated by the above results, backpropagation is a new tool for approximating functions on the basis of examples. The concept of approximating arbitrary L_2 functions using functional forms that do not depend upon either orthogonality or linear superposition (a concept that is known in statistics as nonlinear regression)[1] may well turn out to be an important theme not only in neurocomputing, but in mathematics, engineering,

[1] See [23] for a discussion of the relationship between neurocomputing and statistics.

physics as well; and possibly even in neuroscience (see the Appendix below for a speculation in this direction). The most important results about backpropagation are undoubtedly yet to come.

After presenting an earlier version of this paper (including Theorem 1) at the 1988 INNS Meeting in September 1988, it was brought to my attention that White and his colleagues [32] had independently discovered a similar theorem [32]. Coincidentally, it was White's earlier paper with Gallant [18] that led me to the idea for Theorem 1. Other papers examining the issue of Theorem 1 that have subsequently been brought to my attention include a paper also presented at INNS-88 by Moore and Poggio [46], the 1987 paper of Lapedes and Farber [42], and the 1987 doctoral dissertation of le Cun [44]. Work by Irie and Miyake [34], le Cun [43], and Becker and le Cun [4] is also relevant.

This work was partially supported by the SDI Innovative Science and Technology Program under U. S. Army Strategic Defense Command contract DASGGO-88-C-0112 to HNC, Inc.

Appendix

Until now it has been extremely difficult to believe that the traditional backpropagation neural network architecture was relevant to neurophysiology at the cellular level (however, this is not necessarily true for non-traditional variants of backpropagation such as those of Parker [50,49]). This difficulty follows from the fact that past constructions of the traditional backpropagation architecture have involved non-local processing – which is believed to be impossible in neural tissue [11]. Since the new architecture presented in Section 2 above eliminates the non-locality of backpropagation and makes it a legitimate neural network, while retaining the network's traditional mathematical form, it may be sensible to reexamine the possible biological relevance of backpropagation. As a start in this direction this Appendix presents a plausible, but highly speculative, hypothetical neurophysiological implementation of backpropagation. It is important to point out that the neurons involved in this proposed neural circuit are almost certainly also involved in other circuits as well, some of which might well be active at the same time. Thus, this hypothesis does not attempt to account for the totality of cortical function, merely one aspect of it – namely, the learning of associations or mappings between nearby cortical regions. The hypothesis is attractive in that, as demonstrated in Section 4 above, multilayer networks can learn virtually any desired associative mapping, unlike most of the simple linear associative schemes that have been proposed. (Although, in fairness, some basically linear schemes are more capable than they might initially appear, in particular: the sparse associative memory of Willshaw, et al. [75] as well as the Bidirectional Associative Memory (BAM) of Kosko [40,41] and the Hopfield network [29], at least when the proper capacity improvement modifications [21] are incorporated.)

The hypothesis presented here is that backpropagation is used in the cerebral cortex for learning complicated mappings between nearby areas of cortex that are interconnected by the axons of shallow pyramid neurons. As is well known [15,36], the white matter axons of small to medium shallow pyramids are short and only go to nearby areas of cortex as elements of the short association tracts [13]. The axon collaterals of larger, deeper pyramids also go to these adjacent cortical regions. But, unlike the shallow pyramids, other collaterals of the deep pyramids make up the major brain fascicles that interconnect distant cortical regions and send signals to extracortical areas.

The backpropagation hypothesis assumes that the forward pass of the network is active almost all of the time. The backward pass is triggered only occasionally – namely, when it is necessary to learn something. It is assumed that this learning operation is triggered by some sort of "mismatch detection", attention direction "searchlight", or "important event detection" function carried out by thalamic tissue (including the LGN, MGN, pulvinar, and the thalamus proper) and/or the thalamic reticular complex a la the theories of Grossberg [19,20] and Crick [12]. It is assumed that a special input to cortex from thalamus triggers the backward pass (which then proceeds using Equation [34]). The special thalamic input is assumed to modify the behavior of all of the cells of the effected area of cortex, except the deep pyramids – which are exempted, perhaps by means of the action of horizontal cells (which are known to receive thalamic signals and which preferentially synapse with the apical dendrites of the deep pyramids) [13].

The details of the hypothesis are presented in Figure 5. The deep pyramid cells (which are known to carry out an "integrative" function) are assumed to carry out the feed-forward function of the suns – namely, summing the inputs from the shallow pyramids (which, together with normal stellate cells, are assumed to carry out both the forward and backward pass functions of the planets) and applying a sigmoid function to the sum. Thus, each deep pyramid and the shallow pyramids that feed it make up a sun and planet "unit" of Figures 1 and 2. The feedback function of the suns (summing the back-propagated error signals) is assumed to be carried out by cortical "basket" or "basket stellate" cells which synapse with shallow pyramids over a considerable local area (unlike the normal stellates, which only synapse with a small number of shallow pyramids that are very close) [35]. These basket stellates are assumed to receive input preferentially from the cortico-cortical white matter axons of shallow pyramids of nearby cortical areas (i.e., those participating in the short association tracts of the white matter), but not deep pyramid inputs. The shallow pyramid/normal stellate "planet units" are assumed to receive inputs from deep pyramids exclusively. The supposed operation of this neural circuit is now described. Throughout the discussion it is assumed that the transmitted signals are pulse frequency modulated transmissions ranging continuously between zero firing frequency and some maximum firing frequency (corresponding to the asymptotic 0 and 1 outputs of the backpropagation sigmoid). The need to accommodate negative signals for error feedback is ignored here – presumably some sort of basket stellate or normal stellate/shallow pyramid offset bias could fix this problem.

On the forward pass (i.e., the normal operational association mode) shallow pyramids that have become activated (either by feedforward input from "lower layers", or by external input) send their outputs to a deep

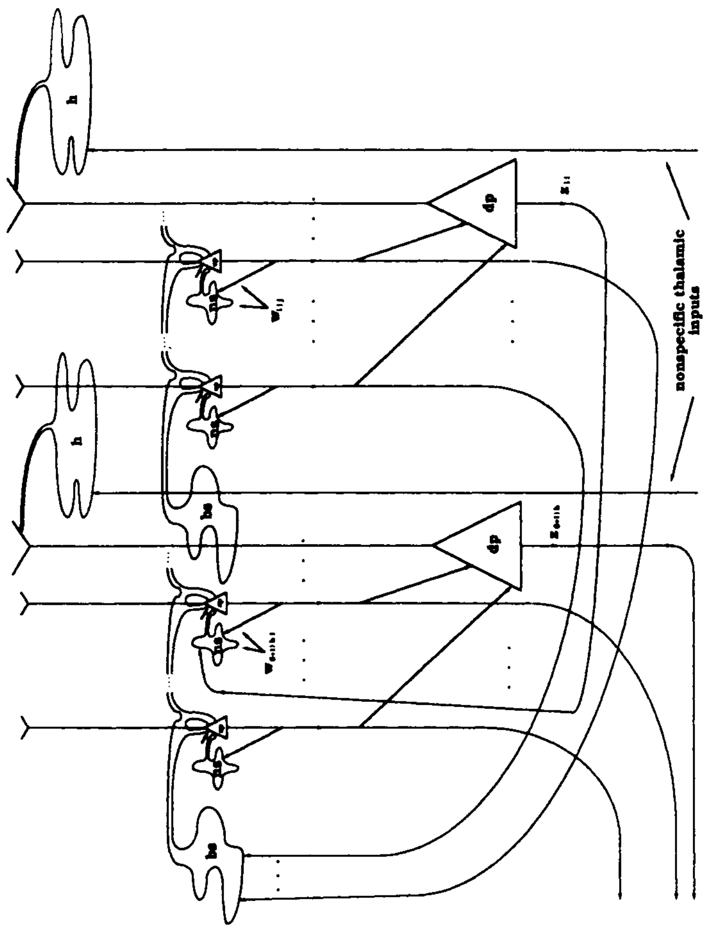

Figure 5: A possible cerebral neurophysiological implementation of backpropagation (essentially Figure 2 done with neurons). In this hypothesis, the deep pyramids (labeled *dp*) carry out the feedforward functions of the suns. Basket stellates (labeled *bs*) carry out the feedback error summation of the suns. Normal stellates (labeled *ns*) and shallow pyramids (labeled *sp*), working together, carry out the functions of the planets (deep pyramid input weighting and weight modification). Thalamic "attention" signals trigger the backward pass (via collaterals not shown) and also activate horizontal cells (labeled *h*) – which allow the deep pyramids to continue firing during learning by exempting them from the influence of the thalamic learning signals.

pyramid that is nearby. Output axons from these shallow pyramids also leave cortex and travel to nearby cortical regions, where they synapse with basket stellates. However, the basket stellates (unlike the normal stellates) are assumed to be inactive until the thalamic learning or "attention" signal is present. The output of the deep pyramid that sums and sigmoids the output of local shallow pyramids is then also sent to nearby cortical areas via the same association bundles containing the shallow pyramid axon collaterals. The deep pyramid axons then synapse with normal stellate/shallow pyramid planet units; thus providing the required forward pass input to the "next layer".

When the thalamus activates the backward pass (which is assumed to only last a short time) the deep pyramids are somehow exempted from interruption of their function. In fact, they keep firing at the same rate they were operating at immediately before the thalamic input occurred. This allows the planet units that receive these inputs to use them in updating their weights by multiplying the incoming deep pyramid signal times the error signal coming in from the basket stellate and adding this to the existing weight. While it is not easy to envision how a couple of cells can carry out these calculations, it does not seem beyond possibility. The shallow pyramids then transmit the product of the basket stellate input (the error for this group of planets) and their weight (either before or after updating). These error signals are then transmitted via the association fascicle to the appropriate basket stellate(s) of the next "lower" layer. Note that in accordance with Figures 1 and 2 that these connections must be very specific (unlike the deep pyramid outputs, which can be broadly distributed). In particular, the shallow pyramid white matter axons that participate in implementation of error backpropagation *must* functionally connect only to the basket cell(s) associated with the shallow pyramids that activate the deep pyramid that feeds that particular shallow pyramid. This does not necessarily mean that each shallow pyramid has only one or a few collaterals. For example, it is possible that the signals in collaterals that target cells other than those few basket stellates needed by backpropagation may be statistically meaningless because they are randomly uncorrelated with activity in the target region. These additional connections might be used to implement other networks at other times. Clearly, the temporal sequencing of events is critical in this hypothesis. In general, we can conclude that for this hypothesis to be correct there must be many more shallow pyramids than deep pyramids and that deep pyramids must have more numerous and more broadly distributed axon collaterals. Clearly, this jibes with the neurophysiological facts. Perhaps this (admittedly crude) cortical backpropagation hypothesis can serve to stimulate some useful thought.

References

1. Amari, S., "A theory of adaptive pattern classifiers", *IEEE Trans. Electronic Computers*, **EC-16** (3), 299–307, 1967.

2. Anderson, James A., and Rosenfeld, Edward, [Eds.], **Neurocomputing: Foundations of Research**, MIT Press, Cambridge, Massachusetts, 1988.

3. Ash, Timur, "Dynamic Node Creation in Backpropagation Networks", Department of Computer Science and Engineering, University of California at San Diego, Preliminary Manuscript, January 1989.

4. Becker, Sue, and le Cun Yann, "Improving the Convergence of Back-Propagation Learning with Second Order Methods", Technical Report CRG-TR-88-5, Connectionist Research Group, University of Toronto, Canada, September 1988.

5. Bronshtein, I.N., and Semendyayev, K.A., **Handbook of Mathematics**, Third Edition, Van Nostrand Reinhold, New York, 1985.

6. Bryson, Arthur E., and Ho, Yu-Chi, **Applied Optimal Control**, Blaisdell, New York, 1969.

7. Carpenter, Gail A., and Grossberg, Stephen, "A Massively Parallel Architecture for a Self-organizing Neural Pattern Recognition Machine", *Computer Vision, Graphics and Image Processing*, **37**, 54-115, 1987.

8. Carpenter, Gail A., and Grossberg, Stephen, "ART 2: self-organization of stable category recognition codes for analog input patterns", *Applied Optics*, **26**, No.23, 4919-4930, 1 December 1987.

9. Carpenter, Gail A., and Grossberg, Stephen, "Associative learning, adaptive pattern recognition, and cooperative-competitive decision making by neural networks", in Szu, Harold (Ed.), **Optical and Hybrid Computing**, SPIE Institute Series, published as: *SPIE Proc.*, **634**, 218-247, 1986.

10. Cater, John, P., "Successfully Using Peak Learning Rates of 10 (and Greater) in Back-Propagation Networks with the Heuristic Learning Algorithm", *Proc. 1987 IEEE International Conference on Neural Networks*, II(645-651), IEEE Press, New York, 1987.

11. Crick, Francis H.C., "The recent excitement about neural networks", *Nature*, **337**, 129-132, 1989.

12. Crick, Francis H.C., "Function of the thalamic reticular complex: the searchlight hypothesis", *Proc. Nat. Acad. Sci.*, **81**, 4586-4590, 1984. { Note: also published in Anderson, James A., and Rosenfeld, Edward, [Eds.], **Neurocomputing: Foundations of Research**, MIT Press, Cambridge, Massachusetts, 1988. }

13. Diamond, Marian C., Scheibel, Arnold B., and Elson, Lawrence M., **The Human Brain Coloring Book**, Barnes & Noble Books, New York, 1985.

14. Dunford, Nelson, and Schwartz, Jacob T., **Linear Operators, Part I**, Third Printing, Wiley Interscience Publishers, New York, 1966.

15. Feldman, Martin L., **Morphology of the Neocortical Pyramidal Neuron** in Peters, Alan, and Jones, Edward G. [Eds.] **Cerebral Cortex, Volume 1: Cellular Components of the Cerebral Cortex**, 123-200, Plenum Press, New York, 1984.

16. Fukushima, Kunihiko, "Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition", *Neural Networks*, **1**, 119-130, 1988.

17. Fukushima, Kunihiko, and Miyake, Sei, "Neocognitron: A New Algorithm for Pattern Recognition Tolerant of Deformations and Shifts in Position", *Pattern Recognition*, **15**, No. 6, 455-469, 1984.

18. Gallant, A. Ronald, and White, Halbert, "There exists a neural network that does not make avoidable mistakes", *Proc. 1988 IEEE International Conference on Neural Networks*, pp I(657-664), IEEE Press, New York, 1988.

19. Grossberg, Stephen [Ed.], **Neural Networks and Natural Intelligence**, MIT Press, Cambridge, 1988.

20. Grossberg, Stephen, **Studies of Mind and Brain**, Reidel, Boston, 1982.

21. Haines, Karen, and Hecht-Nielsen, Robert, "A BAM With Increased Information Storage Capacity", *Proc. 1988 IEEE International Conference on Neural Networks*, I(181-190), IEEE Press, New York, 1988.

22. Hecht-Nielsen, Robert, and Evans, Kellie Michele, "A Method For Error Surface Analysis", *Proc. of NEURONET-90*, Czechoslovakian Academy of Sciences, January 1991.

23. Hecht-Nielsen, Robert, **Neurocomputing**, Addison-Wesley, Reading, Massachusetts, 1990.

24. Hecht-Nielsen, Robert, "On the Algebraic Structure of Feedforward Network Weight Spaces", in: Eckmiller, R. [Ed.], **Advanced Neural Computers**, North-Holland, 129-135, Amsterdam, 1990.

25. Hecht-Nielsen, Robert, "Theory of the backpropagation neural network", *Proc. of the Int. Joint Conf. on Neural Networks*, I, 593–611, IEEE Press, New York, June 1989.

26. Hecht-Nielsen, Robert, "Kolmogorov's Mapping Neural Network Existence Theorem", *Proc. International Conference on Neural Networks*, IEEE Press, New York, III(11-13), 1987.

27. Hinton, Geoffrey E., Sejnowski, Terrence J., and Ackley, David H., **Boltzmann Machines: Constraint Satisfaction Networks that Learn**, Report CMU-CS-84-157, Carnegie-Mellon University, 1984.

28. Hirsch, Morris, "Dynamical Systems Review", a tutorial presented at the 1988 IEEE International Conference on Neural Networks, videotape and notes available from: IEEE Press, New York, 1988.

29. Hopfield, J. J., "Neurons With Graded Response Have Collective Computational Properties Like Those of Two-state Neurons", *Proc. Natl. Acad. Sci.*, **81**, 3088-3092, May 1984.

30. Hornik, Kurt, Stinchcombe, Maxwell, and White, Halbert, "Universal Approximation of an Unknown Mapping and its Derivatives Using Multilayer Feedforward Networks", *Neural Networks*, **3**, 551-560, 1990.

31. Hornik, Kurt, Stinchcombe, Maxwell, and White, Halbert, "Multilayer Feedforward Networks Are Universal Approximators", *Neural Networks*, **2**, 359-366, 1989.

32. Hornik, Kurt, Stinchcombe, Maxwell, and White, Halbert, "Multilayer Feedforward Networks Are Universal Approximators", Manuscript, Department of Economics, University of California at San Diego, June 1988.

33. Hush, D. R., Salas, J. M.,"Improving the Learning Rate of Back-Propagation with the Gradient Reuse Algorithm", *Proc. 1988 IEEE International Conference on Neural Networks*, I(441-446), IEEE Press, New York, 1988.

34. Irie, Bunpei, and Miyake, Sei,"Capabilities of Three-layered Perceptrons", *Proc. 1988 IEEE International Conference on Neural Networks*, I(641-648), IEEE Press, New York, 1988.

35. Jones, Edward G. and Hendry, Stewart H.C., **Basket Cells** in Peters, Alan, and Jones, Edward G. [Eds.] **Cerebral Cortex, Volume 1: Cellular Componants of the Cerebral Cortex**, 309-336, Plenum Press, New York, 1984.

36. Jones, Edward G., **Laminar Distribution of Cortical Efferent Cells** in Peters, Alan, and Jones, Edward G. [Eds.] **Cerebral Cortex, Volume 1: Cellular Componants of the Cerebral Cortex**, 521-553, Plenum Press, New York, 1984.

37. Kohonen, Teuvo, **Self-Organization and Associative Memory**, Second Edition, Springer-Verlag, New York, 1988.

38. Kolmogorov, Andrei Nikolaevich, "On the Representation of Continuous Functions of Many Variables by Superposition of Continuous Functions of One Variable and Addition", Dokl. Akad. Nauk USSR, **114**, 953-956, 1957.

39. Korn, Granino A., and Korn, Theresa M., **Mathematical Handbook for Scientists and Engineers**, Second Edition, McGraw-Hill, New York, 1968.

40. Kosko, Bart, "Bidirectional Associative Memories", *IEEE Trans. on Systems, Man, and Cyber.*, **18**, No. 1, 49-60, January/February 1988.

41. Kosko, Bart, "Adaptive Bidirectional Associative Memories", *Applied Optics*, **26**, No. 23, 4947-4960, 1 December 1987.

42. Lapedes, Alan and Farber, Robert, "How Neural Nets Work", in: Anderson, Dana Z. [ed.], **Neural Information Processing Systems** (Proceedings of the IEEE NIPS conference, Denver, Colorado, 1987), 442-456, American Institute of Physics, New York, 1988.

43. le Cun, Yann, "A Theoretical Framework for Back-Propagation", Technical Report CRG-TR-88-6, Connectionist Research Group, University of Toronto, Canada, September 1988.

44. le Cun, Yann, "Modeles Connexionnistes de l'Apprentissage", Doctoral Dissertation, University of Pierre and Marie Curie, Paris, France, 1987.

45. McInerney, John M., Haines, Karen G., Biafore, Steve, and Hecht-Nielsen, Robert, "Can Backpropagation Error Surfaces Have Non-Global Minima?", Department of Electrical and Computer Engineering, University of California at San Diego, Manuscript, August 1988.

46. Moore, B. and Poggio, T., "Representation Properties of Multilayer Feedforward Networks", Presented at the 1988 Annual Meeting of the International Neural Network Society, Boston, Massachusetts, 8 Sept 1988. Note: an abstract of this paper was published as a Supplement to Volume 1 of *Neural Networks*, 1988.

47. Minsky, Marvin, and Papert, Seymour, **Perceptrons**, Expanded Edition, MIT Press, 1988.

48. Parker, David B.,"Optimal Algorithms for Adaptive Networks: Second Order Back Propagation, Second Order Direct Propagation, and Second Order Hebbian Learning", *Proc. 1987 IEEE International Conference on Neural Networks*, II(593-600), IEEE Press, New York, 1987.

49. Parker, David B.,"A Comparison of Algorithms for Neuron-Like Cells", in Denker, John [Ed.], **Proc. Second Annual Conference on Neural Networks for Computing**, Proceedings Vol. 151, 327-332, American Institute of Physics, New York, 1986.

50. Parker, David B.,"Learning-Logic", Technical Report TR-47, Center for Computational Research in Economics and Management Science, MIT, April 1985.

51. Pineda, F. J., "Recurrent backpropagation and the dynamical approach to adaptive neural computation", *Neural Computation*, **1**, 161–172, Summer 1989.

52. Radons, G., Schuster, H. G., and Werner, D., "Drift and Diffusion in Backpropagation Networks" in Eckmiller, R., Hartmann, G., and Hauske, G. [Eds.] **Parallel Processing in Neural Systems and Computers**, Participants Edition, 261-264, North-Holland, Amsterdam, 1990.

53. Ricotti, L. P., Ragazzini, S., Martinelli, G.,"Learning of Word Stress in a Sub-Optimal Second order Back-Propagation Neural Network", *Proc. 1988 IEEE International Conference on Neural Networks*, I-355 – I-361, IEEE Press, New York, 1988.

54. Rumelhart, David E.,"Parallel Distributed Processing", Plenary Lecture presented at *Proc. 1988 IEEE International Conference on Neural Networks*, San Diego, California, July 1988.

55. Rumelhart, David E., Hinton, Geoffrey E., and Williams, Ronald J., "Learning representations by back-propagating errors", *Nature*, **323**, 533-536, 9 October 1986.

56. Rumelhart, David E., and McClelland, James L., **Parallel Distributed Processing: Explorations in the Microstructure of Cognition**, Vols. I, II & III, MIT Press, 1986 & 1987.

57. Rumelhart, David E., Hinton, Geoffrey E., and Williams, Ronald J., "Learning Internal Representations by Error Propagation", ICS Report 8506, Institute for Cognitive Science, University of California at San Diego, September 1985.

58. Shepanski, J. F.,"Fast Learning in Artificial Neural Systems: Multilayer Perceptron Training Using Optimal Estimation", *Proc. 1988 IEEE International Conference on Neural Networks*, I(465-472), IEEE Press, New York, 1988.

59. Silva, F. M., and Almeda, L. B., "Accelerating Backpropagation", in Eckmiller, R. [Ed.], **Advanced Neural Computers**, Elsevier North Holland, Amsterdam, 1990.

60. Stinchcombe, M., White, H., "Universal Approximation Using Feedforward Networks with Non-Sigmoid Hidden Layer Activation Functions". *Proceedings of the International Joint Conference on Neural Networks*, IEEE Press, 1989.

61. Watrous, Raymond, L., "Learning Algorithms for Connectionist Networks: Applied Gradient Methods of Nonlinear Optimization", *Proc. 1987 IEEE International Conference on Neural Networks*, II(619-627), IEEE Press, New York, 1987.

62. Werbos, Paul J.,"Backpropagation: Past and Future", *Proc. 1988 IEEE International Conference on Neural Networks*, I(343-353), IEEE Press, New York, 1988.

63. Werbos, Paul J. "Building and Understanding Adaptive Systems: A Statistical/Numerical Approach to Factory Automation and Brain Research", *IEEE Trans. on Systems, Man, and Cyber.*, **SMC-17**, No. 1, 7-20, January/February 1987.

64. Werbos, Paul J. "Learning How The World Works: Specifications for Predictive Networks in Robots and Brains", *Proc. 1987 IEEE Conf. on Systems, Man, and Cyber.*, IEEE Press, New York, 1987.

65. Werbos, Paul J., **Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences**, Ph.D. Thesis, Applied Mathematics, Harvard University, November, 1974.

66. White, Halbert, "Connectionist Nonparametric Regression: Multilayer Feedforward Networks Can Learn Arbitrary Mappings", *Neural Networks*, **3**, 535-549, 1990

67. Widrow, Bernard, and Stearns, Samuel D., **Adaptive Signal Processing**, Prentice-Hall, 1985.

68. Widrow, Bernard, "Generalization and Information Storage in Networks of ADALINE Neurons", in Yovitts, G. T., **Self-Organizing Systems**, Spartan Books, 1962.

69. Widrow, Bernard, and Hoff, Marcian T., Jr., "Adaptive Switching Circuits", *IRE WESCON Conv. Record, Part 4*,96-104, 1960.

70. Williams, R. J., and Zipser, D., "A learning algorithm for continually running fully recurrent neural networks", *Neural Computation*, 1, 270–280, Summer 1989.

71. Williams, Ronald J.,"On the Use of Backpropagation in Associative Reinforcement Learning", *Proc. 1988 IEEE International Conference on Neural Networks*, I(623-270), IEEE Press, New York, 1988.

72. Williams, Ronald J.,"A Class of Gradient-Estimating Algorithms for Reinforcement Learning in Neural Networks", *Proc. 1987 IEEE International Conference on Neural Networks*, II(601-608), IEEE Press, New York, 1987.

73. Williams, Ronald J., "Feature Discovery Through Error Correction Learning", ICS Report 8501, Institute of Cognitive Science, University of California at San Diego, May 1985.

74. Williams, Ronald J., "Unit Activation Rules For Cognitive Network Models", ICS Report 8303, Institute of Cognitive Science, University of California at San Diego, November 1983.

75. Willshaw, D.J., Buneman, O.P. and Longuet-Higgins, H.C., "Non-holographic Associative Memory", *Nature*, **222**, 960-962, June 1969.