Pergamon

Neural Networks, Vol. 7, No. 5, pp. 783–796, 1994

0893-6080/94 $6.00 + .00

0893-6080(93)E0024-2

CONTRIBUTED ARTICLE

Perturbation Response in Feedforward Networks

ALI A. MINAI[1] AND RONALD D. WILLIAMS[2]

[1]University of Cincinnati and [2]University of Virginia, Charlottesville

(*Received* 30 *March* 1992; *revised and accepted* 29 *November* 1993)

Abstract—*Feedforward neural networks with continuous-valued activation functions have recently emerged as a powerful paradigm for modeling nonlinear systems. Several classes of such networks have been proved to possess universal approximation capabilities. Prominent among the advantages claimed for such networks are* robustness *and* distributedness *of processing and representation. However, there has been little direct research on either issue, particularly the former, and these characteristics of neural networks have been accepted mostly on faith, or on the basis of heuristic arguments. In this paper, we attempt to construct a framework within which these very important issues can be addressed in a coherent and tractable manner. The focus of the paper is on a particularly simple, but instructive, problem: to predict the effect of perturbations in internal neuron outputs on the performance of the network as a whole. This is directly useful in three ways:* 1) *it gives information about the network's tolerance of internal perturbations;* 2) *it can be used as a criterion for selecting among multiple network solutions to a given modeling problem; and* 3) *it provides a framework for relating the performance of a network to the performance of its components. Of these, the third is especially attractive because it can be used as the basis for a theory of distributed representation and processing in feedforward networks.*

Keywords—Feedforward neural networks, Perturbation, Robustness, Distributed representations, Neuron relevance.

1. INTRODUCTION

Feedforward neural networks represent a very general paradigm for adaptive modeling, pattern recognition, classification, approximation, and inductive learning. Particular interest has recently centered around their ability to induce continuous-valued mappings from limited data, and many researchers have shown that certain broad classes of feedforward neural networks are *universal approximators* (Blum & Li, 1991; Cybenko, 1989; Funahashi, 1989; Hartmann, Keeler, & Kowalski, 1990; Hornik, Stinchcombe, & White, 1988, 1990; Irie & Miyake, 1988; Stinchcombe & White, 1990). This has greatly encouraged the use of such networks in especially difficult modeling problems (e.g., prediction of chaotic sequences and inducing reverse dynamics in adaptive control).

One advantage often claimed for these networks is their *robustness,* that is, their ability to withstand faults, perturbations, lesions, etc. A broad, systematic study of these issues is still lacking, but several researchers have investigated aspects of the problem. A number of these attempts have focused on recurrent networks of binary neurons (Belfore, 1990; Belfore & Johnson, 1989; Belfore, Johnson, & Aylor, 1990; Protzel & Arras, 1990). Stevenson, Winter, and Widrow (1990) have developed a model for predicting the effect of connection weight perturbations in multilayer feedforward networks of binary threshold units (MADALINEs). This work has been extended by Dzwonczyk (1991), who has analyzed the relationship between weight perturbations and the faults that might cause them. Other studies of neural network fault-tolerance include: Bolt (1991a,b,c); Bolt, Austin, and Moran (1992); Segee and Carter (1991); Carter, Rudolph, and Nucci (1990); Qing Xu et al. (1990), etc. Recently, Neti, Schneider, and Young (1990, 1992) have proposed the idea of a *maximally fault-tolerant network* as one that continues to perform satisfactorily after the loss of any single neuron. Detailed, implementation-oriented studies of the reliability of associative memories have been reported by Chung and Krile (1990, 1991). Recently, Bugmann et al. (1992) have proposed direct methods for improving network robustness, and Kerlirzin and

Acknowledgements: This research was supported by the Center for Semicustom Integrated Systems at the University of Virginia and the Virginia Center for Innovative Technology. We would like to thank Prof. Worthy N. Martin and Prof. Barry W. Johnson for their valuable suggestions. A.A.M. would also like to thank the Department of Neurosurgery, University of Virginia, and especially Prof. William B. Levy for their support.

Requests for reprints should be sent to Dr. Ali A. Minai, University of Cincinnati, Electrical and Computer Engineering Dept., Cincinnati, OH 45221; E-mail: ali.minai@uc.edu.

Vallet (1993) have suggested modifications to the back-propagation cost function to enhance robustness in multilayer perceptrons.

The research reported in this paper focuses on the popular class of *approximation networks:* feedforward networks with continuous-valued inputs and outputs, and neurons with continuous-activation functions. The main purpose is to systematically explore the response of such networks to single random perturbations in the outputs of internal neurons. This is useful in two ways:

1. It gives meaningful information about the network's tolerance of internal perturbations, thus providing a useful criterion for selecting among candidate networks with equally good approximation performance on available data sets.
2. It leads to a quantitative framework for relating the performance of the network as a whole to the performance of its components—the neurons.

The first point is obviously important from a practical standpoint, because all optimization methods for approximation networks (e.g., back propagation) lead to nonunique solutions for which quality must be judged on limited data sets. A resistance to internal perturbations is a useful attribute, and its inclusion in the evaluation process could improve the quality of network solutions. It is appropriate to consider perturbations in neurons rather than weights because all faults (including weight faults) ultimately manifest themselves as changes in neuron outputs, and those that do not can be ignored because they have no effect. Conceptually, too, neurons represent the next level below the network as a whole, and are the natural decomposition of the parallel distributed system.

The second point raised above is, however, the more profound. The whole raison d'etre for approximation networks is their *distributedness,* which, it can be claimed, makes them flexible and robust in the first place. Thus, to justify a connectionist solution to a physical modeling problem (as opposed to the modeling of neurological phenomena), it is necessary to give concrete meaning to the claim and advantages of distributedness. In a very obvious sense, distributedness may be seen in terms of the distribution of representational and/or computational responsibility/relevance among the elements of the system—neurons in this case. A model that relates the performance of the system to that of its individual neurons is, therefore, central to a quantitative formulation of such a theory of distributedness, and we see that as an important contribution of the work presented here.

2. NOTATION

A standard network is used throughout this paper. A network, N, consists of an input layer, L_0, a series of $\Omega - 1$ hidden layers, L_k, $1 \leq k \leq \Omega - 1$, and an output layer, L_Ω. The input to layer L_k, $1 \leq k \leq \Omega$, comes entirely from the output of layer L_{k-1}. The number of neurons in L_k is denoted by n_k, with $n_0 \equiv m$ and $n_\Omega \equiv p$. The network has m inputs, x_i, corresponding to the input neurons. The output state of neuron i is denoted by y_i. Input neurons have $y_i = x_i$. Neurons in layer L_k, $k > 0$, have the following characteristic:

$$y_i = \sigma_i(z_i); \quad i \in L_k$$

$$z_i = \phi_i(w_{ij}, \theta_i, y_j); \quad j \in L_{k-1}$$

where w_{ij} is the connection strength from the output of neuron j to neuron i, and θ_i is the bias for i. The function $\sigma_i(\square)$ is called an activation function and can take many forms, for example, sigmoid, Gaussian, etc. The only requirement is that it be twice differentiable everywhere. $\phi(\square)$ is called the composition function, which is usually linear or quadratic. A typical hidden neuron with sigmoid activation has the equation:

$$y_i = \left[1 + \exp\left(-\sum_j w_{ij} y_j - \theta_i\right)\right]^{-1}.$$

Output neurons often have linear activation functions to give them a larger dynamic range.

3. PROBLEM DESCRIPTION

Given the compounded nonlinearities of multilayer approximation networks, only a very simplified setting is considered in this paper. A stochastic perturbation process is defined over individual networks, the aim being to predict the statistics of the change in network output from the parameters of the perturbation process. The main assumptions are:

1. Only one neuron is perturbed at a time.
2. The perturbed neuron is selected randomly from a specified set under a known distribution.
3. The perturbation magnitude is random with a known distribution.
4. Adequate sets of input samples are available.
5. The data sampling, neuron selection, and perturbation magnitude are independent.

The phrase "at a time" means, specifically, "during the presentation of one input vector to the network." Thus, every time a new input is presented, both the location and magnitude of the perturbation change. The purpose is to obtain a *global average characteristic response* of individual networks to random perturbations. This may properly be termed a *(first-order) perturbation response.* The particular case investigated considers perturbations on nonoutput neurons only, and for simple uniform distributions. More complex models can be developed for other distributions by using similar techniques.

4. MEAN OUTPUT DEVIATION

Let N be a network with m inputs, $\{x_1, \ldots, x_m\} \in \mathbf{X} \subseteq \Re^m$, p outputs, $\{o_1, \ldots, o_p\} \in \mathbf{O} \subseteq \Re^p$, and weight

vector W. Let the output of the unfaulted network in response to an input vector $\mathbf{x}_t$ be given by

$$\mathbf{o}_t = N(\mathbf{x}_t; W) = \{o_1(\mathbf{x}_t), \ldots, o_p(\mathbf{x}_t)\}$$

and let the output of the faulted network in response to the same input be

$$\mathbf{o}'_t = N'(\mathbf{x}_t; W) = \{o'_1(\mathbf{x}_t), \ldots, o'_p(\mathbf{x}_t)\}.$$

Then the *deviation* in output o_i due to a perturbation of δ_t on neuron j is defined as

$$\Delta_i(N, \mathbf{x}_t, \delta_t, j, W) = |o'_i(\mathbf{x}_t) - o_i(\mathbf{x}_t)| \tag{1}$$

and the *total output deviation* due to the perturbation as

$$\Delta(N, \mathbf{x}_t, \delta_t, j) = \sum_{i \in L_\Omega} \Delta_i \tag{2}$$

where L_Ω is the set of all output neurons. The principal aim is to get a tractable expression for the expected value of Δ under the assumptions of Section 3. Note that we prefer to use a measure of output error that is defined relative to the *output of the unperturbed network,* and not relative to the ideal desired output given by some arbitrary data set. This, in our opinion, constitutes a relatively objective definition in that it depends only on the network and not on external, subjective desiderata. Of course, a data set is still necessary to calculate the error.

Let $ds(\mathbf{x})$ be an a priori normalized sampling distribution on $\mathbf{X}$, $g(\delta)$ a perturbation distribution defined over $\mathbf{D} \subset \Re$, and $p(j)$ a selection distribution over the set J of all nonoutput neurons. Then, using the independence of $\mathbf{x}$, δ_t, and j, the expected value of Δ_i is given by

$$\langle \Delta_i \rangle = \int_{\mathbf{x}} ds(\mathbf{x}) \int_{\mathbf{D}} g(\delta) d\delta \sum_{j \in \mathbf{J}} p(j) \Delta_i(N, \mathbf{x}, \delta_t, j). \tag{3}$$

If δ_t is sufficiently small, a first-order approximation gives:

$$\Delta_i(N, \mathbf{x}_t, \delta_t, j) \approx \left| \frac{\partial o_i(\mathbf{x}_t)}{\partial y_j(\mathbf{x}_t)} \delta_t \right| \equiv |\mu_{ij}(\mathbf{x}_t)\delta_t|. \tag{4}$$

In the simplest case, when δ is uniformly distributed between $\pm\delta_m$ and $p(j) = |\mathbf{J}|^{-1}\ \forall j$, where $|\mathbf{J}|$ is the cardinality of $\mathbf{J}$, the expected value of Δ_i is estimated as:

$$\langle \hat{\Delta}_i \rangle = \frac{\delta_m |\mathbf{J}|^{-1}}{2} \int_{\mathbf{x}} ds(\mathbf{x}) \sum_{j \in \mathbf{J}} |\mu_{ij}(\mathbf{x})|. \tag{5}$$

Of course, the prior sampling distribution $ds(\mathbf{x})$ is usually not available. Instead, a sufficiently large but finite set of samples $\mathbf{T}$ is used, with the assumption that it is representative of the underlying (possibly application-dependent) sampling distribution. Such a set can be uniformly sampled to estimate $\langle \hat{\Delta}_i \rangle$:

$$\langle \hat{\Delta}_i \rangle = \frac{\delta_m}{2} |\mathbf{J}|^{-1} |\mathbf{T}|^{-1} \sum_{\mathbf{x} \in \mathbf{T}} \sum_{j \in \mathbf{J}} |\mu_{ij}(\mathbf{x})|. \tag{6}$$

This approach is no worse, or better, than the one taken in most nonlinear optimization and estimation algorithms (e.g., in back propagation). Once $\langle \hat{\Delta}_i \rangle$ is calculated, getting $\langle \hat{\Delta} \rangle \equiv \hat{D}(N, \delta_m)$ is straightforward:

$$\hat{D}(N, \delta_m) \equiv \langle \hat{\Delta} \rangle = \frac{\delta_m}{2} |\mathbf{J}|^{-1} |\mathbf{T}|^{-1} \sum_{\mathbf{x} \in \mathbf{T}} \sum_{j \in \mathbf{J}} \sum_{i \in L_\Omega} |\mu_{ij}(x)|, \tag{7}$$

which suggests $r(N) = [|\mathbf{J}|^{-1}|\mathbf{T}|^{-1} \sum_{\mathbf{x} \in \mathbf{T}} \sum_{j \in \mathbf{J}} \sum_{i \in L_\Omega} |\mu_{ij}(\mathbf{x})|]^{-1}$ as a measure of *robustness* for network N over sample set $\mathbf{T}$ and the assumed distributions. As $\mathbf{T}$ grows, r should converge to a *characteristic first-order robustness* $r^*(N) = [|\mathbf{J}|^{-1} \int_{\mathbf{x}} ds(\mathbf{x}) \sum_{j \in \mathbf{J}} \sum_{i \in L_\Omega} |\mu_{ij}(x)|]^{-1}$ for N, and the expected deviation may then be written

$$D(N, \delta_m) \equiv \langle \Delta \rangle \approx \hat{D}(N, \delta_m) = \frac{\delta_m}{2r(N)}. \tag{8}$$

It is important to emphasize that this is only a first-order metric. As such, its applicability depends on: 1) the magnitude of the perturbation (δ); and 2) the curvature of the mapping represented by the network. Conditions necessary for the convergence of the first-order approximation must be met [see Minai (1991) for analysis].

5. VARIANCE OF THE OUTPUT DEVIATION

Although eqn (8) gives an estimate of the expected value of the output deviation Δ in terms of $r(N)$, it is obviously of some interest to estimate the variance of this deviation as well. This can be done using an approach virtually identical to that used estimating D. As before, the derivations assume a uniform probability of neuron faults over $\mathbf{J}$ with one fault at a time, uniform sampling of a representative sample set $\mathbf{T}$, and a perturbation uniformly distributed between $\pm\delta_m$. Analogous results could be derived for other distributions by calculating the sample averages appropriately.

Given the definition (2) of the output deviation, two variances may be calculated. These are:

$$V(N, \delta_m) \equiv \mathrm{Var}[\Delta] = \mathrm{Var}\left[\sum_{i \in L_\Omega} \Delta_i\right] \tag{9a}$$

$$U(N, \delta_m) \equiv \sum_{i \in L_\Omega} \mathrm{Var}[\Delta_i]. \tag{9b}$$

From the definition of variance: $V(N, \delta_m) = \langle \Delta^2 \rangle - \langle \Delta \rangle^2$. Under the uniform distribution assumptions made earlier, it is simple to show that

$$V(N, \delta_m) \approx \hat{V}(N, \delta_m) = \delta_m^2 \left[\frac{\omega(N)}{3} - \frac{1}{4r^2(N)}\right] \tag{10}$$

where $\omega(N)$ is defined by

$$\omega(N) \equiv |\mathbf{J}|^{-1}|\mathbf{T}|^{-1} \sum_{\mathbf{x}\in\mathbf{T}} \sum_{j\in\mathbf{J}} \left[\sum_{i\in L_\Omega} |\mu_{ij}(\mathbf{x})| \right]^2. \quad (11)$$

For $U(N, \delta_m)$, by definition,

$$\begin{aligned} U(N, \delta_m) &= \sum_{i\in L_\Omega} \langle (\Delta_i - \langle \Delta_i \rangle)^2 \rangle \\ &= \sum_{i\in L_\Omega} \langle \Delta_i^2 \rangle - \sum_{i\in L_\Omega} \langle \Delta_i \rangle^2. \end{aligned} \quad (12)$$

As for $V(N, \delta_m)$, it can be shown that

$$U(N, \delta_m) \approx \hat{U}(N, \delta_m) = \delta_m^2 \left[\frac{\chi(N)}{3} - \frac{\psi(N)}{4} \right] \quad (13)$$

where

$$\chi_i(N) \equiv |\mathbf{J}|^{-1}|\mathbf{T}|^{-1} \sum_{\mathbf{x}\in\mathbf{T}} \sum_{j\in\mathbf{J}} |\mu_{ij}(\mathbf{x})|^2 \quad (14)$$

$$\psi_i(N) \equiv |\mathbf{J}|^{-1}|\mathbf{T}|^{-1} \sum_{\mathbf{x}\in\mathbf{T}} \sum_{j\in\mathbf{J}} |\mu_{ij}(\mathbf{x})| \quad (15)$$

$\chi(N) \equiv \sum_{i\in L_\Omega} \chi_i(N)$ and $\psi(N) \equiv \sum_{i\in L_\Omega} \psi_i^2(N)$. Note that

$$\sum_{i\in L} \psi_i(N) = \frac{1}{r(N)} \quad (16)$$

and that the $\psi_i(N)$ add as squares to give $\psi(N)$.

6. BOUNDING ESTIMATION INACCURACY

In using an estimate it may be asked if its inaccuracy can be estimated, or at least bounded? One possible answer can be developed in the framework of power series approximation.

Let o_i be an output of network N, and let y_j be the output of an internal neuron j. Given network input $\mathbf{x}_t \in \mathbf{T}$, the effect of perturbation δ_t in y_j on the network output o_i can be estimated to the second order as:

$$\begin{aligned} o_i' - o_i &\approx \left.\frac{\partial o_i}{\partial y_j}\right|_{\mathbf{x}_t} \delta_t + \frac{1}{2} \left.\frac{\partial^2 o_i}{\partial y_j^2}\right|_{\mathbf{x}_t} \delta_t^2 \\ &= \mu_{ij}(\mathbf{x}_t)\delta_t + \frac{1}{2}\lambda_{ij}(\mathbf{x}_t)\delta_t^2 \end{aligned} \quad (17)$$

where $\lambda_{ij}(\mathbf{x}_t)$ denotes the partial second derivative of o_i with respect to y_j in the perturbation-free network with input $\mathbf{x}_t$. Noting that $|o_i' - o_i| = \Delta_i(N, \mathbf{x}_t, \delta_t, j)$, and using the triangle inequality,

$$\begin{aligned} &|\mu_{ij}(\mathbf{x}_t)\delta_t| - \frac{1}{2}|\lambda_{ij}(\mathbf{x}_t)\delta_t^2| \lesssim \Delta_i(N, \mathbf{x}_t, \delta_t, j) \\ &\approx |\mu_{ij}(\mathbf{x}_t)\delta_t + \lambda_{ij}(\mathbf{x}_t)\delta_t^2| \\ &\lesssim |\mu_{ij}(\mathbf{x}_t)\delta_t| + \frac{1}{2}|\lambda_{ij}(\mathbf{x}_t)\delta_t^2| \end{aligned} \quad (18)$$

where $\lesssim$ means "is approximately less than or equal to" (i.e., up to the validity of the second-order approximation). Simplifying and taking averages over $g(\delta)$ and $p(j)$ gives:

$$-\frac{1}{2}\langle|\lambda_{ij}\delta^2|\rangle \lesssim \langle \Delta_i - \hat{\Delta}_i \rangle \lesssim \frac{1}{2}\langle|\lambda_{ij}\delta^2|\rangle. \quad (19)$$

Defining $l_i(N, \mathbf{T}, g, p)$ as the *absolute expected model error* at output i, as measured over sample set $\mathbf{T}$, with perturbation distribution $g(\delta)$ given as $U[-\delta_m, \delta_m]$, and neuron selection distribution $p(j) = |\mathbf{J}|^{-1}$ $\forall j \in \mathbf{J}$,

$$l_i(N, \mathbf{T}, g, p) = |\langle \Delta_i - \hat{\Delta}_i \rangle| \lesssim \frac{1}{2}\langle|\lambda_{ij}|\rangle\langle\delta^2\rangle, \quad (20)$$

which uses the independence of λ_{ij} and δ. As for $\langle|\mu|\rangle$'s, $\langle|\lambda|\rangle$'s can be estimated empirically:

$$\langle|\lambda_{ij}|\rangle \equiv \varphi_i(N, \mathbf{T}) = |\mathbf{J}|^{-1}|\mathbf{T}|^{-1} \sum_{j\in\mathbf{J}} \sum_{\mathbf{x}\in\mathbf{T}} |\lambda_{ij}(\mathbf{x})|. \quad (21)$$

Because $\langle\delta^2\rangle = (\delta_m^2/3)$,

$$l_i(N, \mathbf{T}, \delta_m) = |\langle \Delta_i - \hat{\Delta}_i \rangle| \lesssim \frac{\delta_m^2 \varphi_i(N, T)}{6}. \quad (22)$$

Defining $l(N, \delta_m) = \sum_{i\in L_\Omega} l_i(N, \mathbf{T}, \delta_m)$ as the overall absolute expected model error, the inaccuracy of the first-order model can be bounded as:

$$l(N, \delta_m) \lesssim \bar{l}(N, \delta_m) = \sum_{i\in L_\Omega} \frac{\delta_m^2 \varphi_i(N, T)}{6} \equiv \frac{\delta_m^2 \varphi(N)}{6}. \quad (23)$$

7. COMPUTING r AND OTHER AVERAGES: THE DYNAMIC FEEDBACK STRATEGY

Clearly, if the robustness model developed above is to be useful, the quantities $r(N)$, $\omega(N)$, $\chi(N)$, $\varphi(N)$, and $\varphi(N)$ should be tractably computable. Essentially, this requires a method to calculate the partial derivatives μ_{ij} and λ_{ij} of each network output with respect to each internal neuron. Fortunately, given the definitions, the *dynamic feedback* method (Werbos, 1974) used by back propagation (Rumelhart, Hinton, & Williams, 1986) can be adapted for this purpose. Specific algorithms for calculating μ_{ij} and λ_{ij} are given in the Appendix. All these derivatives for all neurons can be calculated within the same procedure. If q is the cardinality of the data set used to evaluate robustness, the total number of neuron calls in the procedure is easily shown to be $NC = O(qpW)$, where W is the number of weights in the network. Thus, the algorithm scales linearly in the number of weights and the number of data points. In fact, NC is even lower because the calculation of values for $j \in L_{\Omega-1}$ requires only single reference to L_Ω, not squared reference as implied above. Thus, a tighter value for NC would be:

$$NC = q\left(n_\Omega \sum_{l=0}^{\Omega-2} n_l n_{l+1} + n_{\Omega-1} n_\Omega \right), \quad (24)$$

where n_k is the number of neurons in layer L_k.

8. NEURON RELEVANCE AND DISTRIBUTED REPRESENTATIONS

An important aspect of the model developed above is that it can be used to relate the performance of a network to that of its neurons. Several studies have addressed the issue of determining the *relevance* of neurons (and weights) in a network (Hassibi & Stork, 1993; Le Cun, Denker, & Solla, 1990; Mozer & Smolensky, 1989; Sietsma & Dow, 1988), mainly for the purpose of deleting irrelevant units in the search for a smaller network. We have a somewhat different perspective on the issue.

We define the relevance of a neuron j to the network N as:

$$\rho_j^* = \int_{\mathbf{x}} \sum_{i \in L_a} \left| \frac{\partial o_i(\mathbf{x})}{\partial y_j(\mathbf{x})} \right| ds(\mathbf{x}). \tag{24}$$

As for robustness, an approximation using a realistic, finite data set can be easily calculated with the dynamic feedback method:

$$\rho_j(N) = |\mathbf{T}|^{-1} \sum_{\mathbf{x} \in \mathbf{T}} \sum_{i \in L_a} |\mu_{ij}(\mathbf{x})|. \tag{25}$$

Once the ρ's are available, robustness $r(N)$ is simply the reciprocal of the average neuron relevance:

$$r(N) = \left[|\mathbf{J}|^{-1} \sum_{j \in J} \rho_j(N) \right]^{-1}. \tag{26}$$

Like the relevance measures proposed by other researchers, ρ can be used as part of a pruning algorithm. However, in a sense, pruned, near-minimal networks are contrary to the spirit of distributed models. Pruning attempts to produce networks where each component is *maximally relevant* (in a nonrigorous sense), and some algorithms even go so far as to *concentrate* relevance into the fewest possible components so that the rest can be pruned (Hanson & Pratt, 1989; Karnin, 1990; Weigend, Huberman, & Rumelhart, 1990). In our view, a distributed model should attempt to spread relevance as thinly as possible over the largest available number of components. This ensures that the model is robust and flexible—attributes which are of prime importance in modeling complex and possibly non-stationary data. The main problem with using large networks is poor generalization, but we conjecture that this is caused mostly by using unconstrained training algorithms which allow different components to acquire very different relevance values. Using methods complementary to those used in (Hanson & Pratt, 1989; Weigend, Huberman, & Rumelhart, 1990), and calculating neuron relevances as proposed above, it might be possible to minimize relevance and distribute it evenly over all neurons during training. Conceptually, this can be seen in terms of three imperatives:

1. *The Maximal Irrelevance Imperative* (*MII*): each neuron, on average, must be made as irrelevant as possible.
2. *The Minimal Relevance Imperative* (*MRI*): each neuron must remain as relevant as necessary.
3. *The Uniform Relevance Distribution Imperative* (*URDI*): neuron relevances should be as uniformly distributed as possible.

One way to make these ideas precise would be to define:

$$R(N) = \sum_{j \in \mathbf{J}} \rho_j \tag{27a}$$

$$\bar{\rho}_j = \frac{\rho_j}{R(N)} \tag{27b}$$

$$H_\rho(N) = -\sum_{j \in \mathbf{J}} \bar{\rho}_j \ln \bar{\rho}_j \tag{27c}$$

Then, if $E(N)$ is the approximation error of the network over available data and E^* is the maximum tolerable error, an *optimally robust network* of architecture N is defined as a (possibly nonunique, nonglobal) solution to the following optimization problem:

$$E(N) \le E^* \quad \underset{N \in \mathbf{N}}{\operatorname{argmin}}[R(N)] \quad \underset{N \in \mathbf{N}}{\operatorname{argmax}}[H_\rho(N)] \tag{28}$$

This idea can be seen as generalizing the notion of maximally fault-tolerant networks presented in Neti et al. (1990, 1992). For a detailed discussion of these ideas and possible methods, the reader is referred to Minai (1991). In this paper, we only consider the perturbation model as a means of evaluating neuron relevance in networks—a prerequisite for training networks with distributed relevance. This is tested by calculating a quantity called the mru rate: the fraction of the given network population for which the model correctly identifies the most relevant unit.

9. SIMULATIONS AND RESULTS

The model developed above has been tested on a few relatively simple networks. The basic aim of these tests is to answer the question: *Is the model accurate and useful?* In particular, the simulations address only the very simple model presented in this paper, and make no attempt to go into details of more complicated, more realistic models.

Both trained and randomly generated networks were used to test the model. The former provided a realistic setting for applying the model, but it was computationally expensive to train whole populations of networks. Random networks, on the other hand, were easily generated, but could only be used to test the perturbation model's inherent accuracy without reference to an application.

In all, we report three experiments in this paper: a three-part experiment with trained networks and two

with random ones. In each case, a population of networks was considered. For trained networks, this was generated by training a number of independently initialized networks of the same architecture on an available data set, using a variant of back propagation (Minai & Williams, 1990, 1992). The random networks were generated using a mix of weight ranges from ±10 to ±0.1. There were two distinct phases in each experiment:

1. *Evaluation Phase.* In this phase, a fault-free network was evaluated using the perturbation model with a large, representative data set T of input values. The entire data set was traversed once and the derivative averages (r, ψ, φ, etc.) were computed as described above. These provided a *general characterization* of the network, which could then be used to predict its response to perturbations drawn from various uniform distributions. The model also determined the most relevant unit in each network, using eqn (25).
2. *Testing Phase.* This is the phase that a working perturbation model should render unnecessary. Here, perturbations drawn from a specific 0-mean uniform distribution were introduced into the network as it repeatedly traversed a data set T′, which was distinct from T. The effect of these perturbations on the network's output was monitored and compared with the predictions of the evaluation phase. Because only one neuron was perturbed during one data presentation, it was necessary to go through the data several times, so that almost every combination of input and perturbation site was sampled. Typically, the number of passes over the data set was 20 times the number of perturbable neurons. The generating distributions for T and T′ were assumed to be identical, which reflects the usual real-world situation and is consistent with the model derived above. For all networks, perturbations sites were chosen randomly from the set, J, of nonoutput neurons; the extension to output neurons is trivial.

To test the model's performance relative to perturbation magnitude, several perturbation ranges were used for the uniform perturbation distributions. These were δ_m = 0.1, 0.2, 0.3, 0.4, and 0.5. The network architecture for each experiment is described as $[n_0, n_1(\langle \text{type} \rangle), n_2(\langle \text{type} \rangle), \ldots n_\Omega(\langle \text{type} \rangle)]$, where n_k is the number of neurons in layer L_k, and $\langle \text{type} \rangle$ has value sig or lin depending on whether the neurons in

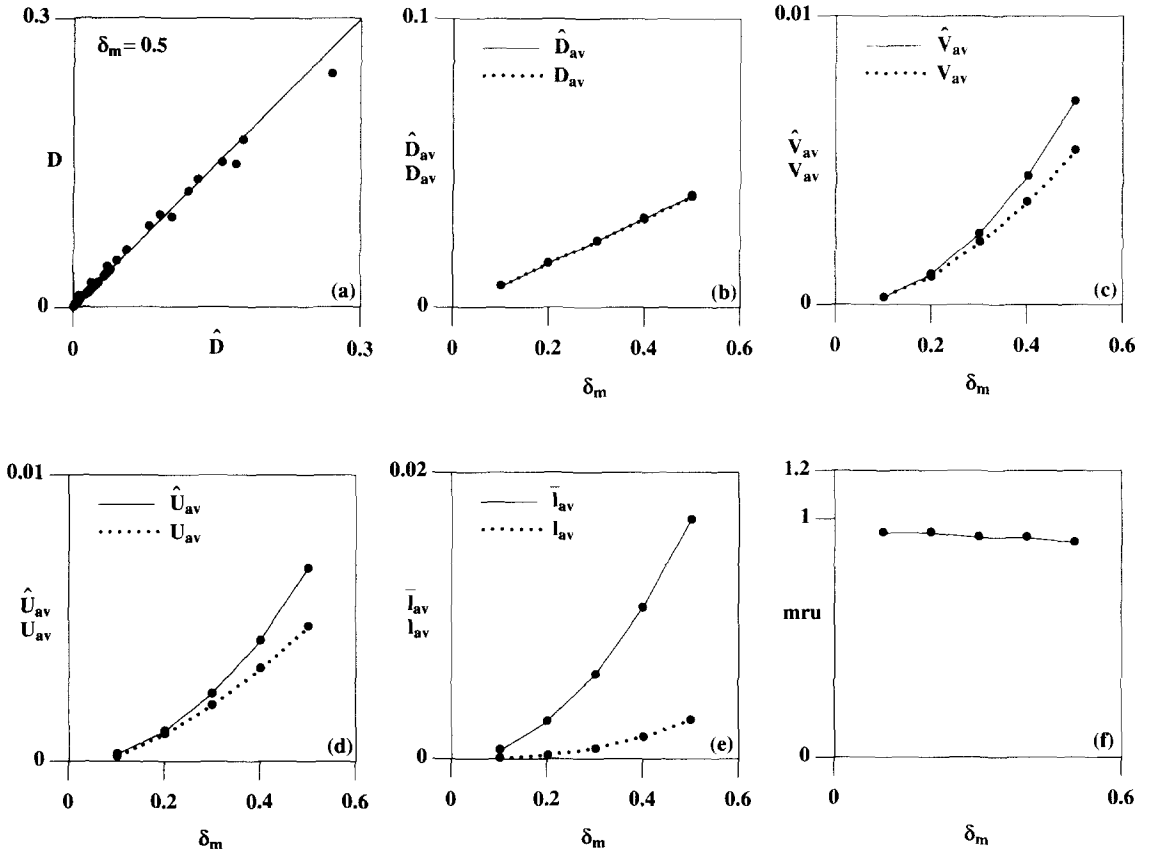

FIGURE 1. Results for 50 randomly generated [2, 8(sig), 8(sig), 2(sig)] networks (Random Network A), using a 1000-point evaluation set and 360 passes over a 1000-point testing set. (a) shows the predicted and actual deviations for δ_m = 0.5; (b), (c) and (d) show the predicted and actual values of *D*, *V* and *U*, respectively; (e) shows the average model error I_{av} and the predicted bound; (f) shows the most relevant unit identification (mru) rate.

the layer are sigmoidal or linear. For each network, the actual most relevant unit was also calculated during the testing phase. This was done by tracking the average output disruption caused by perturbations in each neuron; the neuron to which network output was most sensitive was taken to be the most relevant unit.

10. TARGET ISSUES

In doing the experiments reported here, several issues were specifically targeted. The most important, of course, was the accuracy and applicability of the model, and this was checked by comparing the predictions of the model against actual data from testing. However, there were some deeper issues that also needed to be addressed.

10.1. Heavy Networks

Because the model presented in this paper is based on power-series approximation, it is valid only while the perturbations studied remain within the convergence radii of the various mappings involved. A detailed mathematical analysis of this can be found in Minai (1991), but the key point is simple: the minimal convergence radius for the mapping between an internal neuron output and the network output is inversely related to the magnitude of the weights linking the two. Thus, in networks with large weights, radii of convergence are likely to be relatively small, and the model will be less reliable for large perturbations. It is interesting, therefore, to test networks with large weights (heavy networks) and to compare the results with those for lighter networks.

10.2. Deep Networks

With the possible exception of the output layer, all functional layers in standard approximation networks use simple nonlinear neurons. Thus, the relationship between the output of an internal neuron and an output neuron increases in complexity with the number of intervening layers, and the necessarily local derivatives calculated by the dynamic feedback procedure become less reliable as a basis for extrapolation. This leads to the expectation that networks with several functional layers (deep networks) will prove more difficult for our perturbation model to characterize.

11. SIMULATION 1: RANDOM NETWORK A

In this simulation, the network set consisted of 50 randomly generated [2, 8(sig), 8(sig), 2(sig)] networks,

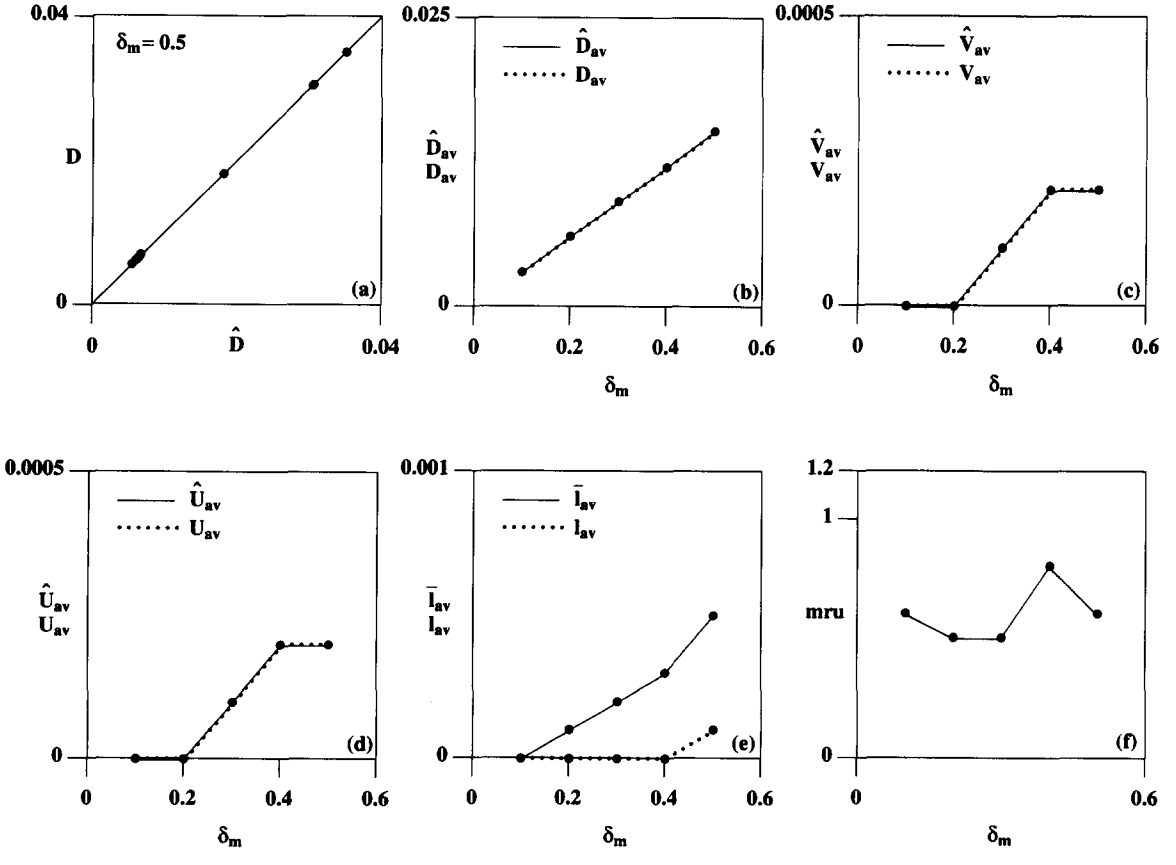

FIGURE 2. Results for 10 randomly generated [2, 200(sig), 2(sig)] networks (Random Network B), five each with weights between ±0.1 and ±1.0, using a 1000-point evaluation set and 500 passes over a 1000-point testing set.

with 10 networks each of weight ranges $w = \{\pm 0.1, \pm 0.5, \pm 1.0, \pm 5.0, \pm 10.0\}$; a network's weights were uniformly distributed in the specified range. Separate simulations were done for each of the five values of δ_m, using a 1000-point evaluation set and 360 passes over a 1000-point testing set. The main purpose here was to evaluate the model on a relatively deep network (three levels of nonlinearity). Figure 1 summarizes the results for the simulations; Figure 1a shows the predicted and actual values of perturbation for $\delta_m = 0.5$, which is the most extreme case; Figure 1b–d show the average predicted and actual values for D, V, and U, respectively; Figure 1e shows the average values of l and $\bar{l}$; and Figure 1f plots the mru rate for each perturbation distribution. In all cases, it is clear that the model was able to accurately predict the perturbation response and its variance, and to effectively bound its own error. The bound, however, is far from tight, and, as in later simulations, appears to be of limited utility.

12. SIMULATION 2: RANDOM NETWORK B

In this two-part experiment, the network architecture was [2, 200(sig), 2(sig)], and the purpose was to test the model on relatively large networks. The first part used 10 networks, five each with weight ranges ±0.1 and ±1.0, and the second part considered five networks with weight range ±10.0, which meant that they were extremely heavy: the output neurons had expected weight vector magnitudes of $5025^{1/2} = 70.88$, which essentially made them threshold units. Both parts used a 1000-point evaluation set and 500 passes over a 1000-point testing set. The results are shown in Figure 2 for the first part and Figure 3 for the second, with the graphs arranged as for Simulation 1.

As expected, the networks in the first set showed much greater resistance to perturbation than those in the second, reflecting the more nonlinear nature of the latter. Also, the performance of the model on the heavier networks was somewhat poorer than on light ones, but the degradation was surprisingly small. This and the previous simulation clearly establish that the model proposed in this paper can adequately handle a variety of useful network architectures.

13. SIMULATION 3: COPPER ELECTRODISSOLUTION TIME SERIES PREDICTION

In this three-part experiment, the networks tested were trained to predict a chaotic time series representing the reaction rate of an electrochemical process (copper

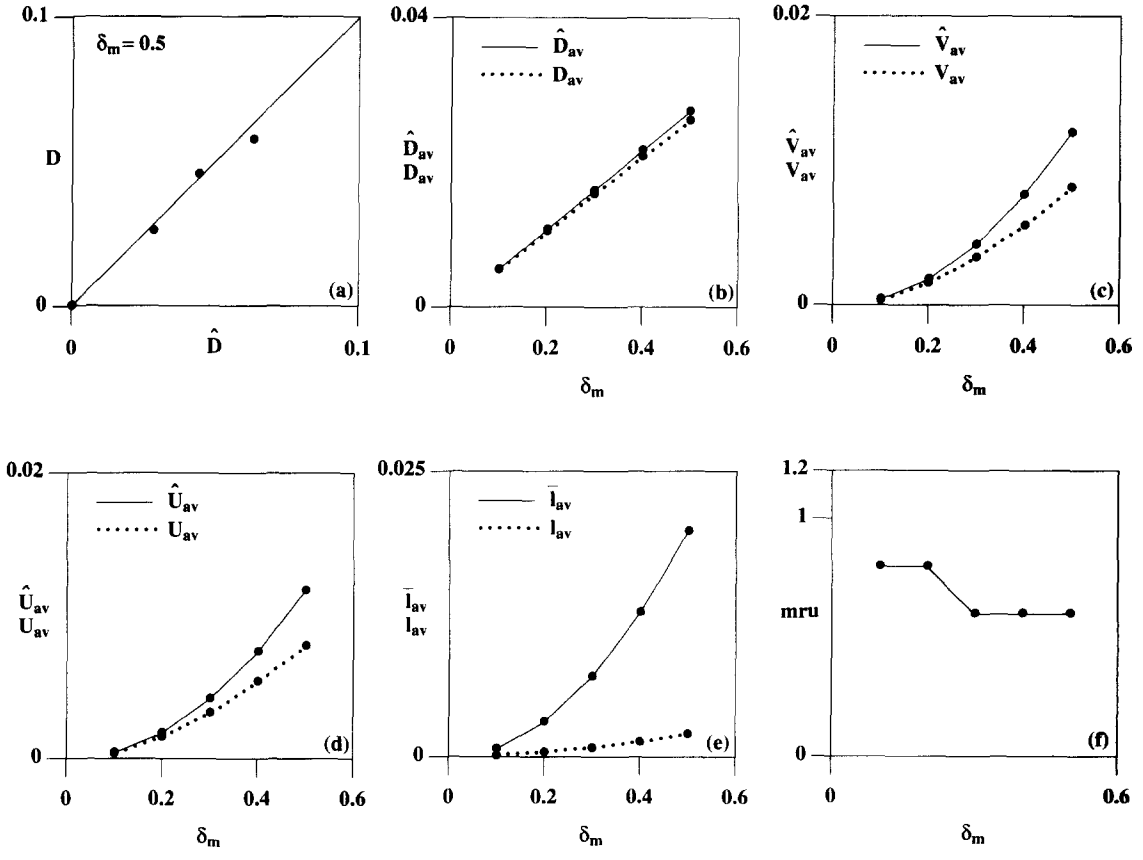

FIGURE 3. Results for five randomly generated [2, 200(sig), 2(sig)] networks (Random Network B) with weights between ±10.0 (heavy networks), using a 1000-point evaluation set and 500 passes over a 1000-point testing set.

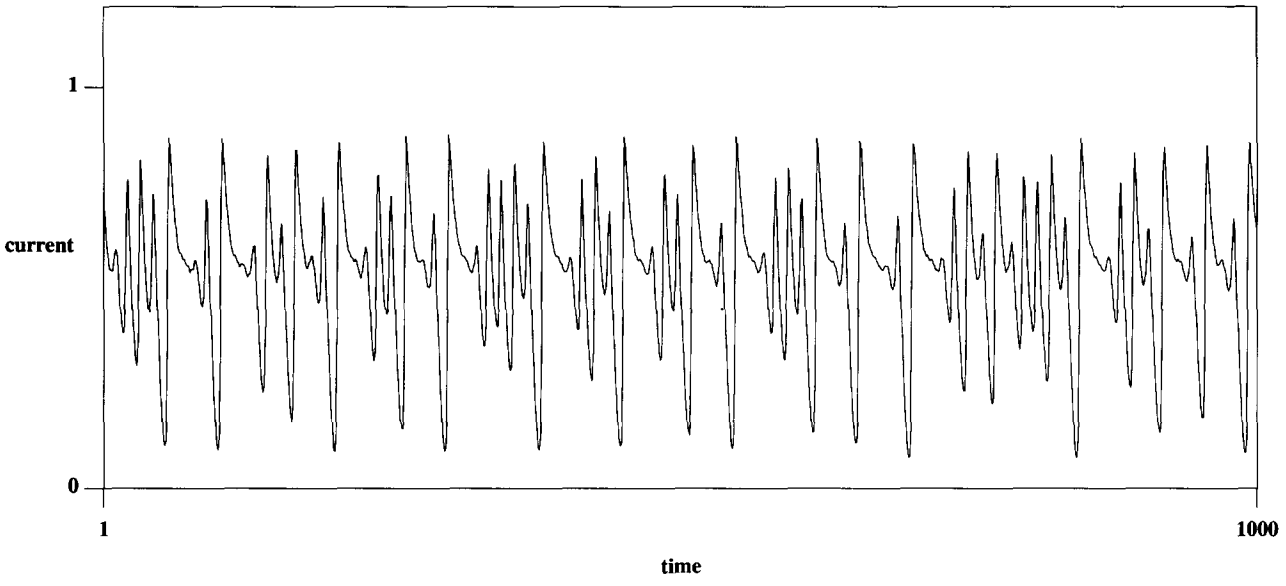

FIGURE 4. Copper electrodissolution time series data used in Simulation 3.

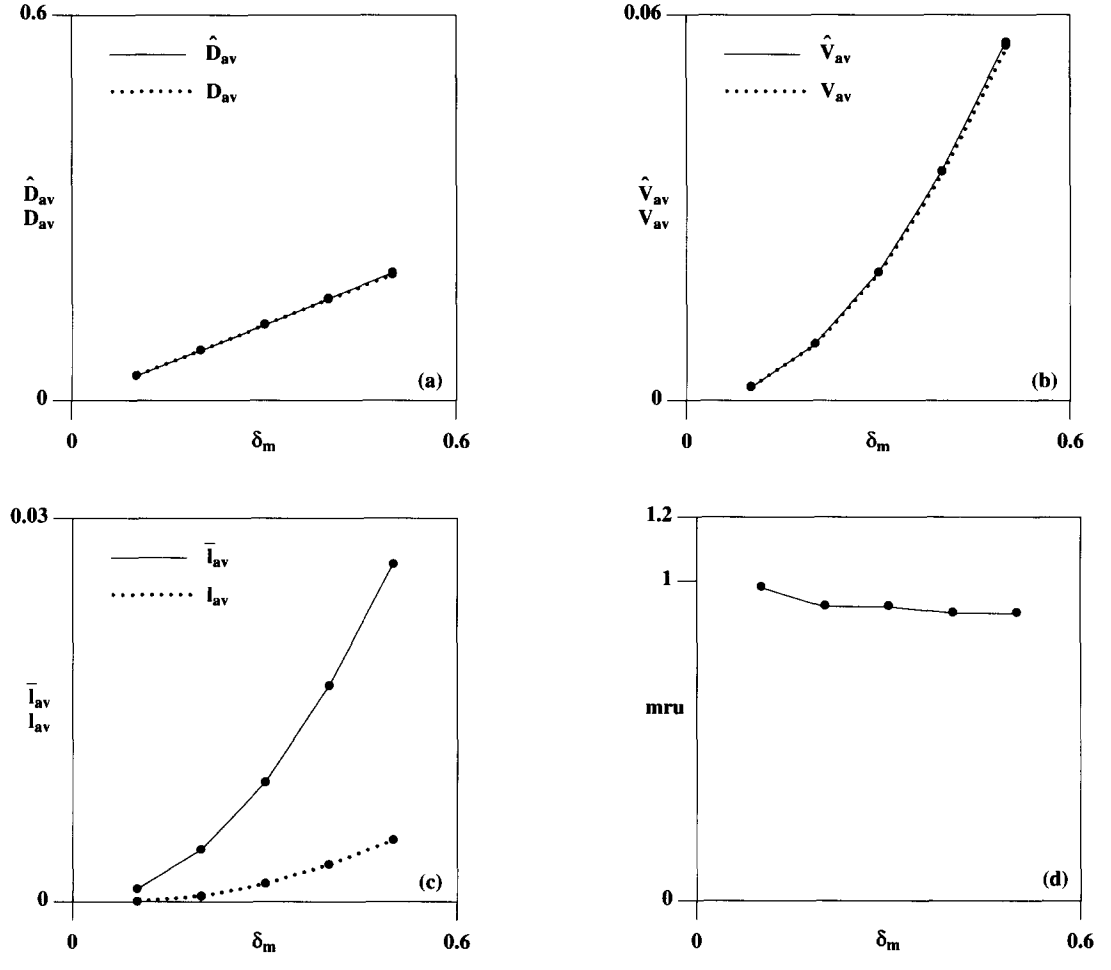

FIGURE 5. Results for 40 [5, 10(sig), 5(lin)] networks, using a 500-point evaluation set and 300 passes over a 500-point testing set. (a) and (b) show the predicted and actual values of *D* and *V*, respectively; (c) shows the average model error I_{av} and the predicted bound; (f) shows the most relevant unit identification (mru) rate.

electrodissolution) (Kube, 1990). The networks used five samples at intervals of 0.005 s to predict the series at the next step. The data were scaled to be between 0 and 1, and are shown in Figure 4. A data set of 800 points was used for training, and another one of 200 to test for generalization. Weights were initialized uniformly between ±1. The results are shown in Figures 5–7. Because all networks in this experiment used a single linear output neuron, U was trivially equal to V, and is not shown in the graphs.

13.1. Part I

The network architecture used in this part was [5, 10(sig), 1(lin)], and 40 networks were evaluated, all with training and generalization errors between 0.01 and 0.02. Although this was not enough for long-range reconstruction of the time series, it represented a reasonable one-step prediction model, and certainly provided a good set of networks for perturbation response evaluation.

As before, the perturbation model was tested with all five values of δ_m. A 500-point set was used for evaluation, and 300 passes over another 500-point set for testing. These sets were obtained by dividing a 1005-point time series into two, which meant that the sampling distributions in both cases reflected real domain-specific considerations.

13.2. Part II

In this part, a larger and deeper network architecture was used to evaluate the effect of these factors. Ten networks with the architecture [(5, 10(sig), 5(sig), 1(lin))] were trained and tested on the data sets used in Part I; the training and testing errors of all networks were between 0.0075 and 0.013, with an average training error of 0.0092 and an average testing error of 0.0093—considerably better than in Part I. Robustness evaluation was done using the 500-point evaluation and testing sets used in Part I, with 400 passes over the testing set.

13.3. Part III

This part used 10 networks with the architecture [5, 10(sig), 10(sig), 1(lin)], trained and tested as in Parts

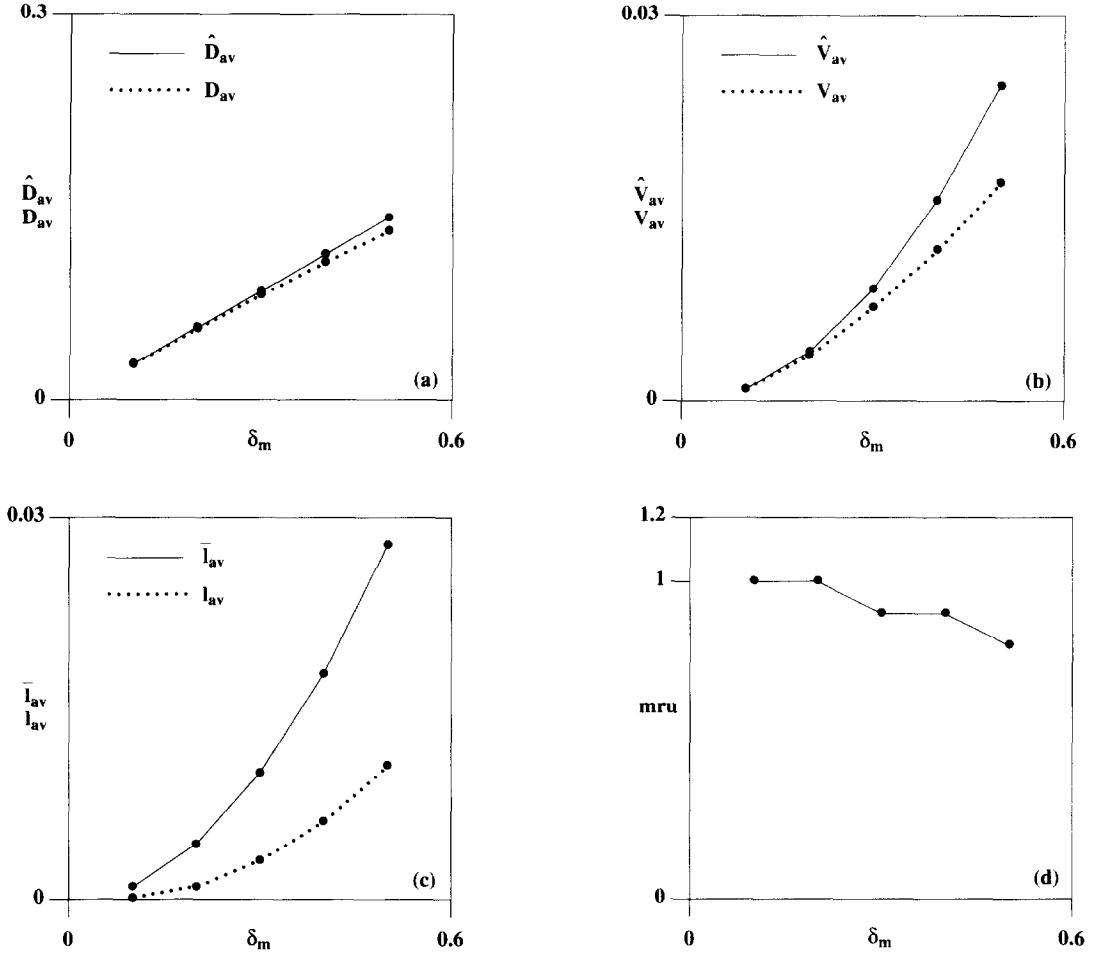

FIGURE 6. Results for 10 [5, 10(sig), 5(sig), 5(lin)] networks, using a 500-point evaluation set and 400 passes over a 500-point testing set.

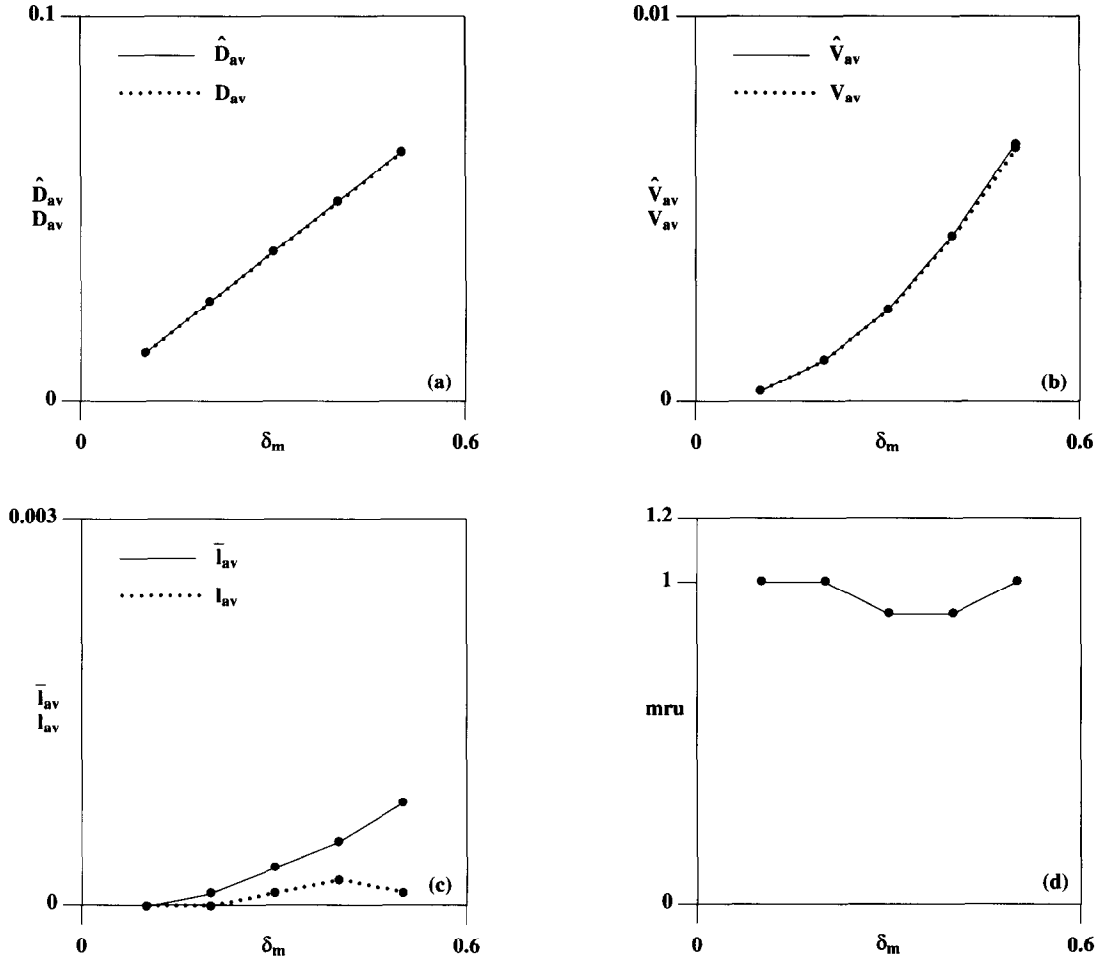

FIGURE 7. Results for 10 [5, 10(sig), 10(sig), 5(lin)] networks, using a 500-point evaluation set and 400 passes over a 500-point testing set.

I and II. The training and testing errors of all networks were between 0.0075 and 0.015, with an average training error of 0.0099 and an average testing error of 0.01, which meant that these networks were comparable to those in Part II. The robustness evaluation and testing used the same 500-point data sets as Parts I and II, with 400 passes over the testing set.

13.4. Discussion of Simulation 3 Results

Simulation 3 was the most interesting of our experiments from a practical standpoint because it involved networks trained on a real approximation problem. As with the random networks, it was clear that the perturbation model performed very adequately in predicting the effect of random perturbations on a variety of network architectures. The good performance on the (5 10 5) network was not surprising because 10 of the 15 perturbable neurons were linearly connected with the output. However, a large number of neuron outputs in the other two architectures passed through one or two levels of nonlinearity, and the good results in those cases were more worthy of remark. Probably the most interesting and significant fact was the success of the model in predicting the most relevant unit. Because performance on this was scored strictly on a success/failure basis, the mru rates of close to 1 reflect an extremely good prediction of neuron relevance in general—especially in the larger architectures. Several studies (e.g., Minai, 1991; Segee & Carter, 1991; Bolt, Austin & Morgan, 1992) have pointed out that standard training algorithms such as back propagation make no provision for distributing computational responsibility/relevance over the set of available neurons. The model presented in this paper provides a quantitative way to assess this. For this to work, the model must be able to predictively determine the relative relevance of individual neurons, and a high mru rate achieved here is an indication of such success.

Another very important point demonstrated by this experiment was that networks with identical architectures, trained on the same data, can show very different perturbation responses. This underscores the usefulness of a perturbation response model that can help select the most perturbation-resistant (robust) network instantiation.

14. IMPLICATIONS FOR GENERALIZATION

Approximation networks are useful only if, having been trained on limited data sets, they can be applied to novel data, that is, if they generalize well. There are some theoretical studies of generalization in feedforward networks (e.g., Schwartz et al., 1990), but recent research has shown that training networks with noise-corrupted data can often improve generalization (e.g., Holmstrom & Koistinen, 1992; Sietsma & Dow, 1991). This is not surprising because, as pointed out in Reed, Oh, and Marks (1992), training with noisy data performs an implicit regularization by specifying target output values over whole neighborhoods rather than at precisely the points in the training set. In essence, the network is forced to become robust and, in the process, comes to generalize better by learning a smoother function. It has also been pointed out by several researchers (e.g., Clay & Sequin, 1992; Sequin & Clay, 1990; Judd & Munro, 1993; Minnix, 1992; Murray & Edwards, 1993) that training with noise improves network fault-tolerance. Interestingly, Sietsma and Dow (1991) also report that networks trained with noisy data *use* more of their neurons than networks trained without noise, that is, the former show a greater degree of distributedness. This bears out the intuitive notion (Minai, 1991) that robust networks with well-distributed computational relevance will also generalize well, because both properties require smoother mappings. As pointed out earlier, this perspective is the converse of the pruning technique, which minimizes units rather than unit participation to improve generalization. The relative merits of these two points of view remain to be studied.

15. CONCLUSION

In this paper, we have presented a model to characterize the response of feedforward neural networks to random perturbations in neuron outputs. We have considered a relatively limited, but interesting, situation. The main motivation behind this was to address the related issues of robustness and distributedness in a clear, tractable way. Both topics—particularly distributedness—are rich research areas, and this research represents only the modest beginnings of a possible framework for such study. Other connectionists have, directly or indirectly, addressed the question of distributedness (e.g., French, 1991; Hinton, McClelland, & Rumelhart, 1986; Smolensky, 1990; van Gelder, 1989, 1991), but there is considerable scope for developing a consistent and powerful theory of distributed representation and processing. Seeing distributedness as a *minimax* assignment of relevance to representational or processing elements seems to us to be a natural definition from a connectionist perspective. Besides giving a concrete and quantitative meaning to the notion, it also relates directly to the other important concepts of generalization and robustness.

REFERENCES

Belfore, L. A., II (1990). *Modeling the performance of faulty neural networks.* Doctoral dissertation, Department of Electrical Engineering, University of Virginia, Charlottesville, VA.

Belfore, L. A., II, & Johnson, B. W. (1989). The fault tolerance of neural networks. *International Journal of Neural Network Research and Applications,* **1,** 24–41.

Belfore, L. A., II, Johnson, B. W., & Aylor, J. H. (1990). The modeling of fault-tolerance in neural networks. *Proceedings of the International Joint Conference on Neural Networks, Washington, DC,* **I,** 325–328.

Bishop, C. (1992). Exact calculation of the hessian matrix for the multilayer perceptron. *Neural Computation,* **4,** 494–501.

Blum, E. K., & Li, L. K. (1991). Approximation theory and feedforward networks. *Neural Networks,* **4,** 511–516.

Bolt, G. R. (1991a). Fault models for artificial neural networks. *Proceedings of the International Joint Conference on Neural Networks, Singapore,* **III,** 1918–1923.

Bolt, G. R. (1991b). Fault tolerance of lateral interaction networks. *Proceedings of the International Joint Conference on Neural Networks, Singapore,* **II,** 1373–1378.

Bolt, G. R. (1991c). Assessing the reliability of artificial neural networks. *Proceedings of the International Joint Conference on Neural Networks, Singapore,* **I,** 578–583.

Bolt, G. R., Austin, J., & Morgan, G. (1992). *Fault tolerant multilayer perceptrons* (Tech. Rep. YCS 180). York, UK: University of York, Computer Science Department.

Bugmann, G., Sojka, P., Reiss, M., Plumbley, M., & Taylor, J. G. (1992). Direct approaches to improving the robustness of multilayer neural networks. In I. Aleksander & J. G. Taylor (Eds.), *Artificial Neural Networks II: Proceedings of the International Conference on Artificial Neural Networks, Brighton, UK.* Oxford: Elsevier Science Publishers B.V.

Carter, M. J., Rudolph, F. J., & Nucci, A. J. (1990). Operational fault tolerance of CMAC networks. In D. S. Touretzky (Ed.), *Advances in neural information processing systems* 2. San Mateo, CA: Morgan Kaufmann.

Chung, Pao-Choo, & Krile, T. F. (1990). Reliability measures for hebbian-type associative memories with faulty interconnections. *Proceedings of the International Joint Conference on Neural Networks, San Diego, CA,* **I,** 847–852.

Chung, Pao-Choo, & Krile, T. F. (1991). Reliability characteristics of hebbian-type associative memories in network implementations. *Proceedings of the International Joint Conference on Neural Networks, Seattle, WA,* **II,** 363–368.

Clay, R. D., & Sequin, C. H. (1992). Fault-tolerance training improves generalization and robustness. *Proceedings of the International Joint Conference on Neural Networks, Baltimore, MD,* **I,** 769–774.

Cybenko, G. (1989). Approximations by superpositions of a single function. *Mathematics of Control, Signals and Systems,* **2,** 303–314.

Dzwonczyk, M. J. (1991). *Quantitative failure modes of feed-forward neural networks.* MS thesis, Massachusetts Institute of Technology, Cambridge, MA (also Tech. Rep. CSDL-T-1068, The Charles Stark Draper Laboratory, Inc.).

French, R. M. (1991). Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks (Tech. Rep. 51-1991). Bloomington, IN: Indiana University, Center for Research on Concepts and Cognition.

Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks,* **2,** 183–192.

Hanson, S. J., & Pratt, L. Y. (1989). Comparing biases for minimal

network construction with back-propagation. In D. S. Touretzky (Ed.), *Advances in neural information processing systems* 1 (pp. 177–185). San Mateo, CA: Morgan Kaufmann.

Hartmann, E., Keeler, J. D., & Kowalski, J. M. (1990). Layered neural networks with Gaussian hidden units as universal approximators. *Neural Computation,* **2,** 210–215.

Hassibi, B., & Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. In S. J. Hanson, J. D. Cowan, & C. L. Giles (Eds.), *Advances in neural information processing systems* 5 (pp. 164–171). San Mateo, CA: Morgan Kaufmann.

Hinton, G. E., McClelland, J. L., & Rumelhart, D. E. (1986). Distributed representations. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (pp. 77–109). Cambridge, MA: MIT Press.

Holmstrom, L., & Koistinen, P. (1992). Using additive noise in back-propagation training. *IEEE Transactions on Neural Networks,* **3,** 24–38.

Hornik, K., Stinchcombe, M., & White, H. (1988). *Multi-layer feedforward networks are universal approximators* (Tech. Rep.). La Jolla, CA: University of California at San Diego, Department of Economics.

Hornik, K., Stinchcombe, M., & White, H. (1990). Using multilayer feedforward networks for universal approximation. *Neural Networks,* **3,** 551–560.

Irie, B., & Miyake, S. (1988). Capabilities of three-layered perceptrons. *Proceedings of the IEEE International Conference on Neural Networks, San Diego, CA,* **I,** 641–648.

Judd, S., & Munro, P. W. (1993). Nets with unreliable hidden nodes learn error-correcting codes. In S. J. Hanson, J. D. Cowan, & C. L. Giles (Eds.), *Advances in neural information processing systems* 5 (pp. 89–96). San Mateo, CA: Morgan Kaufmann.

Karnin, E. D. (1990). A simple procedure for pruning back-propagation trained neural networks. *IEEE Transactions on Neural Networks,* **1,** 239–242.

Kerlirzin, P., & Vallet, F. (1993). Robustness in multilayer perceptrons. *Neural Computation,* **5,** 473–482.

Kube, M. C. (1990). *Time series prediction using backpropagation neural networks.* Masters thesis, Department of Chemical Engineering, University of Virginia, Charlottesville, VA.

Le Cun, Y., Denker, J. S., & Solla, S. (1990). Optimal brain damage. In D. Touretzky (Ed.), *Advances in neural information processing systems* 2 (pp. 598–605). San Mateo, CA: Morgan Kaufmann.

Minai, A. A. (1991). The robustness of feed-forward neural networks: a preliminary investigation. Doctoral dissertation, University of Virginia, Charlottesville, VA.

Minai, A. A., & Williams, R. D. (1990). Back-propagation heuristics: A study of the extended delta-bar-delta algorithm. *Proceedings of the International Joint Conference on Neural Networks, San Diego, CA,* **I,** 676–679.

Minai, A. A., & Williams, R. D. (1992). Fast back-propagation with adaptive decoupled momentum. In B. Soucek (Ed.), *Fast learning and invariant object recognition* (pp. 9–26). New York: John Wiley & Sons, Inc.

Minnix, J. I. (1992). Fault-tolerance of the backpropagation neural network trained on noisy inputs. *Proceedings of the International Joint Conference on Neural Networks, Baltimore, MD,* **I,** 847–852.

Mozer, M. C., & Smolensky, P. (1989). Skeletonization: A technique for trimming the fat from networks via relevance assessment. In D. S. Touretzky (Ed.), *Advances in neural information processing systems* 1 (pp. 107–115). San Mateo, CA: Morgan Kaufmann.

Murray, A. F., & Edwards, P. J. (1993). Synaptic weight noise during MLP learning enhances fault-tolerance, generalisation and learning trajectory. In S. J. Hanson, J. D. Cowan, & C. L. Giles (Eds.). *Advances in neural information processing systems* 5 (pp. 491–498). San Mateo, CA: Morgan Kaufmann.

Neti, C., Schneider, M. H., & Young, E. D. (1990). Maximally fault tolerant neural networks and nonlinear programming. *Proceedings of the International Joint Conference on Neural Networks, San Diego, CA,* **II,** 483–496.

Neti, C., Schneider, M. H., & Young, E. D. (1992). Maximally fault tolerant neural networks. *IEEE Transactions on Neural Networks,* **3,** 14–23.

Protzel, P. W., & Arras, M. K. (1990). Fault-tolerance of optimization networks: Treating faults as additional constraints. *Proceedings of the International Joint Conference on Neural Networks, Washington, DC,* **I,** 455–458.

Qing, Xu, Jurgens, C., Arrue, B., Minnix, J., Johnson, B., & Inigo, R. M. (1990). A fault-tolerance analysis of the neocognitron model. *Proceedings of the International Joint Conference on Neural Networks, Washington, DC,* **II,** 559–562.

Reed, R., Oh, S., & Marks, R. J., II (1992). Regularization using jittered training data. *Proceedings of the International Joint Conference on Neural Networks, Baltimore, MD,* **III,** 147–152.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (pp. 318–362). Cambridge, MA: MIT Press.

Schwartz, D. B., Samalam, V. K., Solla, S. A., & Denker, J. S. (1990). Exhaustive learning. *Neural Computation,* **2,** 371–382.

Segee, B. E., & Carter, M. J. (1991). Fault-tolerance of pruned multilayer networks. *Proceedings of the International Joint Conference on Neural Networks, Seattle, WA,* **II,** 447–452.

Sequin, C. H., & Clay, R. D. (1990). Fault-tolerance in artificial neural networks. *Proceedings of the International Joint Conference on Neural Networks, San Diego, CA,* **I,** 703–708.

Sietsma, J., & Dow, R. J. F. (1988). Neural net pruning—why and how? *Proceedings of the IEEE Conference on Neural Networks, San Diego, CA,* **I,** 325–332.

Sietsma, J., & Dow, R. J. F. (1991). Creating artificial neural networks that generalize. *Neural Networks,* **4,** 67–79.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence,* **46,** 159–216.

Stevenson, M., Winter, R., & Widrow, B. (1990). Sensitivity of feedforward neural networks to weight errors. *IEEE Transactions on Neural Networks,* **1,** 71–80.

Stinchcombe, M., & White, H. (1990). Universal approximation using feedforward networks with non-sigmoid hidden layer activation function. *Proceedings of the International Joint Conference on Neural Networks, Washington, DC,* **I,** 613–617.

van Gelder, T. (1989). *Distributed representations.* Doctoral dissertation, University of Pittsburgh, Pittsburgh, PA.

van Gelder, T. (1991). What is the "D" in "PDP"? A survey of the concept of distribution. In S. Stich, W. Ramsay, & D. Rumelhart (Eds.), *Philosophy and connectionist theory* (pp. 33–59). Hillsdale, NJ: Lawrence Erlbaum Associates.

Weigend, A. S., Huberman, B. A., & Rumelhart, D. E. (1990). Predicting the future: A connectionist approach. *International Journal of Neural Systems,* **1,** 193–209.

Werbos, P. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences.* Doctoral dissertation, Harvard University, Cambridge, MA.

APPENDIX

Algorithm for Calculating μ_{ij}

Consider the network output o_i and a neuron $j \in L_{\alpha-1}$ with output y_j. Clearly, for a given $\mathbf{x}_l \in \mathbf{T}$,

$$\mu_{ij} = \frac{\partial o_i}{\partial y_j} = \frac{\partial o_i}{\partial z_i}\frac{\partial z_i}{\partial y_j}. \tag{A1}$$

If the neuron i has a sigmoidal activation function and $z_i = \sum_{j \in L_{\Omega-1}} w_{ij} y_j + \theta_i$,

$$\mu_{ij} = w_{ij} o_i (1 - o_i) \tag{A2a}$$

and if i has a linear activation function ($o_i = z_i$),

$$\mu_{ij} = w_{ij}. \tag{A2b}$$

Thus, μ_{ij} can be calculated for all neurons in layer $L_{\Omega-1}$ with respect to all output neurons.

Once all neurons in $L_{\Omega-1}$ have their μ's, neurons belonging to the next lower layer can be considered. Let $j \in L_{\Omega-2}$. Dynamic feedback can be used recursively to find μ_{ij}. Because it is assumed that direct connections exist only between adjacent layers, we obtain

$$\mu_{ij} = \frac{\partial o_i}{\partial y_j} = \sum_{k \in L_{\Omega-1}} \frac{\partial o_i}{\partial y_k} \frac{\partial y_k}{\partial y_j} = \sum_{k \in L_{\Omega-1}} \frac{\partial o_i}{\partial y_k} \frac{\partial y_k}{\partial z_k} \frac{\partial z_k}{\partial y_j}$$

$$= \sum_{k \in L_{\Omega-1}} \mu_{ik} \frac{\partial y_k}{\partial z_k} \frac{\partial z_k}{\partial y_j}. \tag{A3}$$

If all k have sigmoid activation functions, eqn (A3) gives:

$$\mu_{ij} = \sum_{k \in L_{\Omega-1}} \mu_{ik} w_{kj} y_k (1 - y_k). \tag{A4}$$

Because all μ_{ik} have already been calculated, and y_k and w_{kj} are available, calculating μ_{ij} is straightforward. Thus, μ_{ij}'s can be calculated recursively for all $i \in L_\Omega$ and all internal neurons j.

Algorithm for Calculating λ_{ij}

The procedure to calculate the partial second derivatives λ_{ij} is exactly the same as that for μ_{ij}. However, the calculations differ somewhat. The equations for the $j \in L_{\Omega-1}$ case and the $j \notin L_{\Omega-1}$ case are derived below. In each case, an output neuron i is considered.

Case I ($j \in L_{\Omega-1}$). By definition

$$\lambda_{ij} = \frac{\partial^2 o_i}{\partial y_j^2} = \frac{\partial}{\partial y_j} \left[\frac{\partial o_i}{\partial z_i} \frac{\partial z_i}{\partial y_j} \right] = \frac{\partial o_i}{\partial z_i} \frac{\partial^2 z_i}{\partial y_j^2} + \left(\frac{\partial z_i}{\partial y_j} \right)^2 \frac{\partial^2 o_i}{\partial z_i^2}. \tag{A5}$$

If z_i is a linear combination, $(\partial^2 z_i)/(\partial y_j^2) = \partial/(\partial y_j)\,[\partial z_i/\partial y_j] = (\partial w_{ij})/(\partial y_j) = 0$. Thus, eqn (A5) reduces to

$$\lambda_{ij} = \left(\frac{\partial z_i}{\partial y_j} \right)^2 \frac{\partial^2 o_i}{\partial z_i^2}. \tag{A6}$$

If the activation function of i is a sigmoid, $(\partial^2 o_i)/(\partial z_i^2) = (1 - o_i)^2 o_i - o_i^2 (1 - o_i)$, which gives

$$\lambda_{ij} = w_{ij}^2 [(1 - o_i)^2 o_i - o_i^2 (1 - o_i)]. \tag{A7}$$

Case II ($j \notin L_{\Omega-1}$). Suppose $j \in L_A$. Then, applying the dynamic feedback rule:

$$\frac{\partial^2 o_i}{\partial y_j^2} = \frac{\partial}{\partial y_j} \left[\sum_{k \in L_{A+1}} \frac{\partial o_i}{\partial y_k} \frac{\partial y_k}{\partial y_j} \right] = \sum_{k \in L_{A+1}} \frac{\partial}{\partial y_j} \left[\frac{\partial o_i}{\partial y_k} \frac{\partial y_k}{\partial y_j} \right]. \tag{A8}$$

Now

$$\frac{\partial}{\partial y_j} \left[\frac{\partial o_i}{\partial y_k} \frac{\partial y_k}{\partial y_j} \right] = \frac{\partial o_i}{\partial y_k} \frac{\partial^2 y_k}{\partial y_j^2} + \frac{\partial^2 o_i}{\partial y_k \partial y_j} \frac{\partial y_k}{\partial y_j}$$

$$= \frac{\partial o_i}{\partial y_k} \frac{\partial^2 y_k}{\partial y_j^2} + \frac{\partial^2 o_i}{\partial y_k^2} \left(\frac{\partial y_k}{\partial y_j} \right)^2 = \mu_{ik} \frac{\partial^2 y_k}{\partial y_j^2} + \lambda_{ik} \left(\frac{\partial y_k}{\partial y_j} \right)^2. \tag{A9}$$

Because all the terms on the right-hand side of this equation are known, each term in the summation of eqn (A8) can be calculated. Specifically, if k has a sigmoid activation function, eqn (A9) becomes

$$\frac{\partial}{\partial y_j} \left[\frac{\partial o_i}{\partial y_k} \frac{\partial y_k}{\partial y_j} \right] = \mu_{ik} w_{kj}^2 [(1 - y_k)^2 y_k - y_k^2 (1 - y_k)]$$

$$+ \lambda_{ik} (w_{kj} y_k (1 - y_k))^2. \tag{A10}$$

μ_{ik} and λ_{ik} have already been calculated by the backward propagation procedure (because k belongs to the layer above j). Thus, eqn (A8) for internal neuron j is:

$$\frac{\partial^2 o_i}{\partial y_j^2} \sum_{k \in L_{A+1}} \{ \mu_{ik} w_{kj}^2 [(1 - y_k)^2 y_k - y_k^2 (1 - y_k)]$$

$$+ \lambda_{ik} (w_{kj} y_k (1 - y_k))^2 \}. \tag{A11}$$

A more comprehensive procedure for calculating arbitrary second derivatives in feedforward network networks is given by Bishop (1992).